

audiokinetic

Wwise

User's Guide

2015.1.9



Wwise

User's Guide

Wwise 2015.1.9 Revision 1910
Copyright © 2016 Audiokinetic Inc.

This document (whether in written, graphic or video form) is supplied as a guide for the Wwise® product. This documentation is the property of Audiokinetic Inc. (“Audiokinetic”), and protected by Canadian copyright law and in other jurisdictions by virtue of international copyright treaties. It may be used by you in accordance with the following.

This documentation may be duplicated, reproduced, stored or transmitted, exclusively for your internal, non-commercial purposes, but you may not alter the content of any portion of the documentation. Any copy of the documentation shall retain all copyright and other proprietary notices contained therein.

The content of this documentation is furnished for information purposes only, and its content is subject to change without notice. Reasonable care has been taken in preparing the information contained in this document, however, we disclaim all representations, warranties and conditions, whether express, implied or arising out of usage of trade or course of dealing, concerning this documentation and assume no responsibility or liability for any losses or damages of any kind arising out of the use of this guide or of any error or inaccuracy it may contain, even if we have been advised of the possibility of such loss or damage.

Wwise®, Audiokinetic®, Actor-Mixer®, SoundFrame® and SoundSeed® are registered trademarks, and Master-Mixer™, SoundCaster™ and Randomizer™ are trademarks, of Audiokinetic. Other trademarks, trade names or company names referenced herein may be the property of their respective owners.

Table of Contents

Welcome to Wwise 2015.1.9	xi
I. Where to Begin?	1
1. Wwise Roadmap	3
Where to Begin	4
Guide to the Wwise Roadmap	4
Document Conventions	4
Wwise Help	5
Contacting Support	7
2. Getting Started	8
Starting and Exiting Wwise	9
Exploring the Wwise Interface	10
Wwise Interface Basics	12
Personalizing Your Workspace	23
Speeding Up the Way You Work	38
II. Setting Up Your Projects	50
3. Working with Projects	52
Overview	53
Managing Projects	55
Defining your Project Settings	60
Defining the Default User Settings for your Project	88
Troubleshooting Your Project	90
The Learning Annex - More on Obstruction and Occlusion	96
Project Management Tips and Best Practices	98
4. Managing Platforms	100
Overview	101
Platform Manager	101
PS Vita Hardware Versus Software Platforms	104
5. Managing Workgroups	107
Overview	108
Dividing Your Project into Work Units	111
Viewing the Status of Your Project Files	120
Using Wwise with Your Source Control System	122
Resolving Project Inconsistencies	123
Managing Project Files Using a Workgroup Plug-in	125
Workgroup Tips & Best Practices	138
6. Managing Media Files in Your Project	142
Overview	143
The Importing Process	143
Importing Media Files	151
Replacing Media Files	167
Managing File Import Issues	170
Re-Organizing Media Files within the Originals Folder	172

Clearing Your Cache	173
Editing Audio Files in an External Editor	174
Creating Audio and Motion Sources Using Plug-ins	176
Media File Management Tips and Best Practices	177
7. Building Your Actor-Mixer Hierarchy	178
Overview	179
About Properties in the Project Hierarchy	185
Building the Actor-Mixer Hierarchy	194
Enhancing Sound and Motion by Randomizing Property Values	200
Building Actor-Mixer Hierarchy Tips and Best Practices	201
8. Building the Structure of Output Busses	206
Overview	207
Defining the Properties of a Bus	212
Structuring a Bus Hierarchy - Example	217
III. Using Sounds and Motion to Enhance Gameplay	220
9. Defining Object Playback Behaviors	222
Overview	223
Defining the Playback Behavior for Sound and Motion Objects	223
Defining the Playback Behavior for Random/Sequence Containers	230
Defining the Contents and Behavior of Switch Containers	242
Defining the Contents and Behavior of the Blend Container	248
Object Playback Tips & Best Practices	259
10. Defining Positioning for Sound and Motion	261
Overview	262
Understanding Positioning in Wwise	263
Working with 2D Sound, Music, and Motion FX Objects	267
Working with 3D Sound, Music, and Motion FX Objects	269
Applying Distance-Based Attenuation	271
Defining Spatial Positioning Using Animation Paths	287
Routing Audio Signals to the Center Speaker	298
Positioning Tips and Best Practices	299
Understanding Channel Configurations	306
Speakers vs Headphones Panning Rules	308
11. Managing the Priority of Sounds and Motion	310
Overview	311
Understanding How Wwise Prioritizes Sounds and Motion Objects	313
Limiting Object Playback Instances	316
Defining Playback Priority	320
Managing Low-Volume Sounds and Motion Objects	323
Priority Tips and Best Practices	325
12. Managing Effects	329

Overview	330
Using Effects	330
Using Effects to Implement Environment Acoustics	340
Effects Tips and Best Practices	344
13. Managing Motion	346
Overview	347
Understanding How Motion Works in Wwise	347
Creating Motion for Your Game	350
Building an Output Structure for Motion	358
Motion Tips and Best Practices	359
IV. Interacting with the Game	362
14. Managing Events	364
Overview	365
Creating Events	370
Working with Events	378
Event Tips and Best Practices	382
15. Managing Dynamic Dialogue	384
Overview	385
Understanding the Dynamic Dialogue System	385
Creating Dialogue Events	389
Working with Dialogue Events	400
Dialogue Events Tips and Best Practices	404
16. Working with States	407
Overview	408
Working with States	409
Defining Transitions Between States in a State Group	413
Assigning States to Objects and Busses	414
States Tips and Best Practices	420
17. Working with Switches	421
Overview	422
Working with Switches	423
Mapping Game Parameter Values to Switches	426
Switches Tips and Best Practices	427
18. Working with RTPCs	429
Overview	430
Managing Game Parameters Used in RTPCs	431
Controlling Property Values Using Game Parameters	436
Working with LFOs	443
Working with Envelopes	445
Viewing Game Objects	447
RTPC Tips & Best Practices	447
19. Working with Triggers	449
Overview	450
Working with Triggers	451
20. Working with States and State Groups for Dynamic Dialogue	453

Overview	454
Working with State Groups	455
V. Creating Interactive Music	458
21. Understanding Interactive Music	461
Overview	462
Understanding Interactive Music	462
Interactive Music Tips & Best Practices	464
22. Building the Interactive Music Hierarchy	467
Overview	468
What is a Music Segment?	468
Types of Containers	470
About Properties in the Interactive Music Hierarchy	471
Adding Objects to the Interactive Music Hierarchy	472
Adding Parent Objects	474
Managing Music Objects in the Interactive Music Hierarchy	476
Building Interactive Music Hierarchies Tips & Best Practices	477
23. Defining Music Object Playback Behaviors	478
Overview	479
Defining the Time Settings for Music Objects	479
Defining the Playback Behavior of Music Playlist Containers	480
Defining the Contents and Behaviors of Music Switch Containers	484
Interactive Music Playback Tips & Best Practices	488
24. Working with Music Tracks and Segments	490
Overview	491
Adding Music Tracks to Segments	493
Defining the Playback Behavior for Music Tracks	494
Adding Sub-tracks to Tracks	497
Associating Sub-tracks to Switches/States	497
Populating Tracks	498
Removing Tracks and Sub-tracks from Segments	499
Snapping to Increments in the Music Editor	499
Working with Clips	500
Auditioning Segments	502
Working with Cues	504
25. Working with MIDI	508
Creating MIDI Content	509
Importing MIDI files	509
Understanding MIDI content and MIDI target	509
Mixing MIDI and Audio contents	511
Understanding MIDI Tempo	511
Changing the Playback Speed of MIDI	512
26. Creating MIDI Instruments	513
Designing a Synth One Instrument	514
Designing a Simple Sampled MIDI Instrument	514

Understanding MIDI Note Tracking	515
Understanding the MIDI Filters	515
Understanding the MIDI Events	516
Adding Fade-in and Fade-out on MIDI events	517
Using MIDI Data to Control Object Property Values	517
Using the MIDI Keymap Editor	518
Testing an Instrument with a MIDI Keyboard	518
Routing MIDI from a DAW to Wwise	519
27. Working with Transitions	521
Overview	522
Understanding Transitions	523
Adding Transitions	524
Copying and Pasting Transitions	525
Removing Transitions	526
Setting Source and Destination Properties	526
Using Transition Segments	530
Interactive Music Transitions Tips & Best Practices	532
28. Using Stingers	534
Overview	535
Adding Stingers	536
Defining Playback Settings for Stingers	538
Removing Stingers	539
Auditioning Stingers	540
VI. Finishing Your Project	542
29. Managing Output	545
Overview	546
Specifying the Output Routing for Sound, Music, and Motion Objects	547
Using the User-Defined Auxiliary Sends	549
Using the Game-Defined Auxiliary Sends	550
Using Loudness Normalization or Make-up gain to Adjust Volume	551
Understanding the Voice Pipeline	552
Understanding Secondary Outputs	554
Understanding HDR	558
Using HDR	562
More about HDR	571
Creating the Final Mix	576
30. Managing Platform and Language Versions	594
Overview	595
Authoring Across Platforms	595
Localizing Your Project	620
Versions Tips and Best Practices	628
31. Creating Simulations	631
Overview	632

Building a Simulation	633
Managing Playback of Your Simulation	637
Simulating with Game Syncs	640
Fine-Tuning Properties in a Simulation	644
Creating Simulations Tips and Best Practices	648
32. Managing Memory in Wwise	649
Overview	650
Understanding the Components of the Memory Manager	650
Setting the Size of Your Memory Pools	651
Troubleshooting Memory Problems	654
Optimizing Memory Pools	654
Memory Management Tips and Best Practices	660
33. Profiling	662
Overview	663
Understanding the Different Types of Profiling in Wwise	664
Connecting to a Local/Remote PC or Game Console	669
Capturing Data from the Sound Engine	672
Monitoring and Troubleshooting with the Performance Monitor	683
Keeping Track of Objects and Listeners with the Game Object Explorer	686
Examining Objects with the Game Object 3D Viewer	691
Evaluating Game Syncs with the Game Sync Monitor	696
Profiling Tips and Best Practices	697
34. Managing SoundBanks	699
Overview	700
Understanding How SoundBanks are Loaded in a Game	702
Building SoundBanks	707
Managing SoundBanks	725
Defining Custom Attributes for Your SoundBanks	728
Generating SoundBanks for a Project	738
Using the CopyStreamedFiles Tool	743
Strategies for Managing SoundBanks	744
SoundBanks Tips and Best Practices	759
35. Managing File Packages	761
Overview	762
Working with File Packager Projects	762
Managing File Packages within a Project	765
Downloadable Content Overview	770
Generating File Packages	772
Using File Packager Arguments in the Command Line	773
File Packager Tips and Best Practices	777
VII. Working with Wwise	779
36. Getting to Know the Project Explorer	782
Overview	783

Visual Elements in the Project Explorer	785
Working in the Project Explorer	786
37. Getting to Know the Event Viewer	789
Overview	790
Working with the Event Viewer	791
38. Getting to Know the Property Editor	796
Overview	797
Working with the Property Editor	804
39. Getting to Know the Contents Editor	808
Overview	809
Working with the Contents Editor	816
40. Getting to Know the Transport Control	825
Overview	826
Setting Playback Properties	828
Pinning an Object in the Transport Control	831
Playing/Pausing/Stopping Content	832
Using Game Syncs During Playback	833
41. Getting to Know the Schematic View	838
Overview	839
Customizing the Schematic View	839
Working with the Schematic View	841
42. Getting to Know the Graph View	845
Overview	846
Changing the Display of the Graph View	847
Working with Control Points in the Graph View	852
Working with Curves in the Graph View	855
43. Getting to Know the Timeline	859
Overview	860
Working with the Timeline for Positioning	863
Working with the Music Segment Editor Timeline	864
44. Working with Searches, Queries, and References	866
Overview	867
Searching for Elements within Your Project	867
Finding the Project Elements that Reference a Particular Object	870
Working with Queries	872
Queries - Tips and Best Practices	880
45. Using Presets	882
Overview	883
Using Presets	883
46. Using a Control Surface	887
Overview	888
Connecting a Control Surface Device to Wwise	888
Creating a Control Surface Session	889
Understanding Control Surface Bindings	889

Creating Control Surface Bindings	890
Understanding Control Surface View Groups	893
Handling Conflicts in Control Surface Sessions	895
Using the Control Surface Toolbar	895
VIII. Appendix	897
A. Regular Expression Quick Reference Guide	899
B. Shortcuts	901
Zoomable Editor	902
Attenuation Editor	902
Audio File Management	902
Contents Editor	903
Game Object 3D Viewer	903
Game Profiler	904
Global - Contextual (Active view and selected object)	904
Music Segment Editor	904
Position Editor (User-defined)	905
Project Explorer	905
RTPC Graph View	905
Schematic View	905
Soundcaster	906
Transport Control	906
Glossary	907

Welcome to Wwise 2015.1.9

Welcome to Wwise® version 2015.1.9, the middleware solution from Audiokinetic that gives you the power to create great audio and motion for video games. Through the tight integration of an advanced authoring application, a robust sound engine, and a thorough SDK to control them both, Wwise can increase your productivity and enhance your creative output.

Built to address the specific needs of the game development pipeline, Wwise is a unique solution for designers, composers, and programmers. By allowing you to develop game audio, music, and motion concurrently with game visuals, Wwise facilitates the design and authoring of sophisticated audio and motion during every phase of game development.

About This Guide

This guide is designed for designers, integrators, scriptwriters, composers, and programmers working in the game development industry. It assumes that you have general computer and audio knowledge. Some basic concepts are explained, but only in relation to the Wwise software. This guide gives you specific information and step-by-step instructions for the most common tasks that you will use in Wwise.

To learn more about its organization, see the [Wwise Help](#) page.

Before you Begin

Before working with Wwise, you may want to read the Wwise Fundamentals document to gain a better understanding of the key concepts and workflow of Wwise. After reviewing the Wwise Fundamentals document, you can then start reading the individual chapters in each part of the Help. These chapters provide conceptual and procedural information, examples, and tips and best practices for using Wwise.

Using Other Wwise Documents and Support

The Wwise Help is only one component in our comprehensive set of materials and resources that are available for you. Consult the [Guide to the Wwise Roadmap](#) for information about these other guides and resources.

audiokinetic

Part I. Where to Begin?



1. Wwise Roadmap	3
Where to Begin	4
Guide to the Wwise Roadmap	4
Document Conventions	4
Wwise Help	5
Contacting Support	7
2. Getting Started	8
Starting and Exiting Wwise	9
Exploring the Wwise Interface	10
Wwise Interface Basics	12
Personalizing Your Workspace	23
Speeding Up the Way You Work	38

Chapter 1. Wwise Roadmap












Where to Begin	4
Guide to the Wwise Roadmap	4
Document Conventions	4
Wwise Help	5
Contacting Support	7

Where to Begin

Audiokinetic has provided you with a comprehensive set of materials and resources. Before you begin using Wwise, you should consult the roadmap to fully understand what is available to you. We hope that these materials and resources will help you along the way as you integrate Wwise into your production pipeline.

Guide to the Wwise Roadmap

Refer to the following table for a complete guide to the different materials and resources available with Wwise.

	Release Notes	Refer to the Release Notes for new feature information, product limitations, and workarounds.
	Wwise Installation and Migration Guide	Use the Installation and Migration Guide to install or reinstall Wwise and its component applications and migrate your projects from one version to another.
	Wwise Video Tutorials	Watch video tutorials , available on our website for all users, to learn Wwise concepts and how to perform specific tasks in Wwise.
	Wwise Online Help	Consult the online help for on-screen information about all the options and views in Wwise.
	Wwise User Guide	Refer to the User Guide for basic and advanced task-based information.
	Wwise Quick Reference	Print the Quick Reference to have a list of shortcuts handy.
	Wwise Game Simulator Help	Refer to the Game Simulator Help for information about how to create scripts to test game scenarios on various platforms.
	Wwise SDK documents	Refer to the SDK documentation for information about integrating the sound engine , SoundFrame® , plug-ins, and more.
	Wwise Sample Project	Review the sample project and its corresponding documentation for an in-depth look at real-world sound design examples in Wwise.
	Audiokinetic Q&A	Access our Audiokinetic Q&A to ask and answer questions within a community of Wwise users and experts.
	Wwise Support Center	Access our online support center .

Document Conventions

Information in all Wwise user documentation is displayed using the following conventions:

- [Visual Identifiers](#)

- [Keyboard and Mouse Conventions](#)

Visual Identifiers

The following icons help to identify certain types of information:



Note

Notes are used to provide important additional information.



Tip

Tips are useful bits of information, workarounds, and shortcuts that you may find helpful in a particular situation.



Caution

Cautions are used when you can lose or damage information, such as deleting data or not being able to easily undo an action. Cautions always appear before you are about to do such a task.

Keyboard and Mouse Conventions

Wwise takes advantage of the left, center, and right mouse buttons. Unless otherwise stated, use the left mouse button. The following table shows the terms relating to the mouse and keyboard.

This term	Means this with a mouse
Click	Quickly press and release the left mouse button. Always use the left mouse button unless otherwise stated.
Right-click	Quickly press and release the right mouse button.
Double-click	Click the left mouse button twice rapidly.
Shift+click, Ctrl+click, Alt+click	Hold down the Shift, Ctrl, or Alt key as you click a mouse button.
Drag	Hold down the left mouse button while you move the mouse. This is equivalent to drag and drop.
Alt+key, Ctrl+key, Shift+key	Hold down the first key while you press the second key. For example, “Press Ctrl+Z” means to hold down the Ctrl key while you press the Z key.

Wwise Help

Wwise Help is a context-sensitive help system that contains both reference topics describing each view, field, and option in Wwise, and task-based

information. The documentation is divided into twelve separate parts (nine in the Wwise User's Guide) to provide you with comprehensive and easy-to-find information about Wwise.

- **Part I, “Where to Begin?”** - Includes an overview of the Wwise documentation set, an introduction to many of the concepts in Wwise and getting started information to get you up and running with Wwise quickly, and some helpful information on organizing your workspace and working efficiently with Wwise.
- **Part II, “Setting Up Your Projects”** - Includes conceptual and procedural information on how to set up a project, either alone or within a workgroup environment, how to define or adjust platforms for your project, how to deal with the assets in your project, and how to build and get the most out of the Wwise hierarchy of assets.
- **Part III, “Using Sounds and Motion to Enhance Gameplay”** - Includes conceptual and procedural information on how to enhance the rich and immersive game environments using a variety of properties and behaviors, positioning settings, playback priority, effects, and motion.
- **Part IV, “Interacting with the Game”** - Includes conceptual and procedural information on how to use events, dynamic sequences, and Wwise game syncs to drive the sounds in your game.
- **Part V, “Creating Interactive Music”** - Includes an overview and introduction to the Interactive Music concepts in Wwise, along with extensive procedural information on how to create interactive music for your game.
- **Part VI, “Finishing Your Project”** - Includes information on how to create the final mix, troubleshoot and simulate different aspects of your project, and generate the SoundBanks and file packages for your game.
- **Part VII, “Working with Wwise”** - Describes the functionality of specific Wwise views to help you get up and running quickly with the software.
- **Wwise Reference (Not included in Wwise User's Guide)** - Provides complete information about all the options and views in Wwise, and is accessible through the contextual help.
- **Wwise Source/Effect Plug-ins (Not included in Wwise User's Guide)** - Provides descriptions of each option for the source and effect plug-ins that ship with Wwise.
- **Wwise Tools (Not included in Wwise User's Guide)** - Provides information about various tools that ship with Wwise, including SoundFrame and the Multi-Channel Creator.
- **Part VIII, “Appendix”** - Provides supplementary reference information on regular expression use in Wwise and on Wwise default keyboard shortcuts.
- **Glossary** - Provides an alphabetical list of Wwise and audio-related terms with corresponding definitions.

Related Topics

- [Setting the Documentation Preferences](#)

Accessing Wwise Context-Sensitive Help

When you are looking for information about a specific option or view in Wwise, simply open the Help from that view and read the corresponding information.

To access Wwise context-sensitive Help:

1. Do one of the following:

Click the Help icon in the upper right corner of any view or dialog box.

Press F1.

Wwise Help opens displaying information about the view or dialog box you are currently in.

Accessing Wwise Help

To review the entire Help system, you can open the Wwise Help start page and navigate to different topics from there.

To open Wwise Help:

1. From the menu bar, click **Help** > **Wwise Help**.

Wwise Help opens on the Start Page.

Contacting Support

Use the Audiokinetic [Customer Portal](#) to submit a support ticket.



Tip

You can also ask the Wwise user community through the [Q&A](#) of our website.

Chapter 2. Getting Started

Starting and Exiting Wwise	9
Exploring the Wwise Interface	10
Wwise Interface Basics	12
Personalizing Your Workspace	23
Speeding Up the Way You Work	38

Starting and Exiting Wwise

Wwise runs on Microsoft Windows® XP, Vista, 7, and 8, as well as Mac OS X. To give you quick access to Wwise, a shortcut is automatically added to your desktop when you install Wwise.

To start Wwise:

1. Do one of the following:

Double-click the Wwise icon on your desktop.

From the Windows Start menu, select **Wwise {version} (64-bit)**.

If it is the first time that you run Wwise, the End-User License Agreement opens.

2. Read the agreement and, if you accept the terms, click **Accept**.

The **Project Launcher** window opens.

3. Click **New**.

The **New Project** dialog box opens.

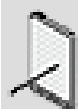
4. In the **Name** field, type the name of your project.
5. To change the location where your project files will be saved, do the following:

Click the **Browse** button (...).

The **Browse For Folder** dialog box opens.

Navigate to the folder where you want Wwise to store your project files.

Click **OK**.



Note

If you want to create a new folder specifically for your audio files, click the **Make New Folder** button.

6. To change the directory where your original audio files will be saved, do the following:

Click the **Browse** button (...).

The **Browse For Folder** dialog box opens.

Navigate to the folder where you want Wwise to store your original copies of the audio files imported into your project.

Click OK.

7. Click OK.

Wwise opens with a new project.

Exiting Wwise

After you have completed a work session, save your work and exit the application.

To exit Wwise:

1. From the menu bar, click **Project > Exit**.

Wwise closes the project and exits.



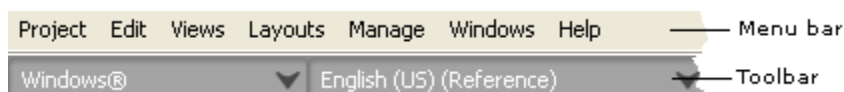
Note

If you haven't already saved your project, Wwise will prompt you to do so before exiting.

Exploring the Wwise Interface

The Wwise interface is divided into several different views. Each view has a specific purpose and gives you access to a series of tools or options that help you manage and define the audio and motion content for your game. Views are grouped together to create layouts, which facilitate the work involved for a particular task or job. There are many different layouts available in Wwise. Refer to [Working with Layouts](#) for more information.

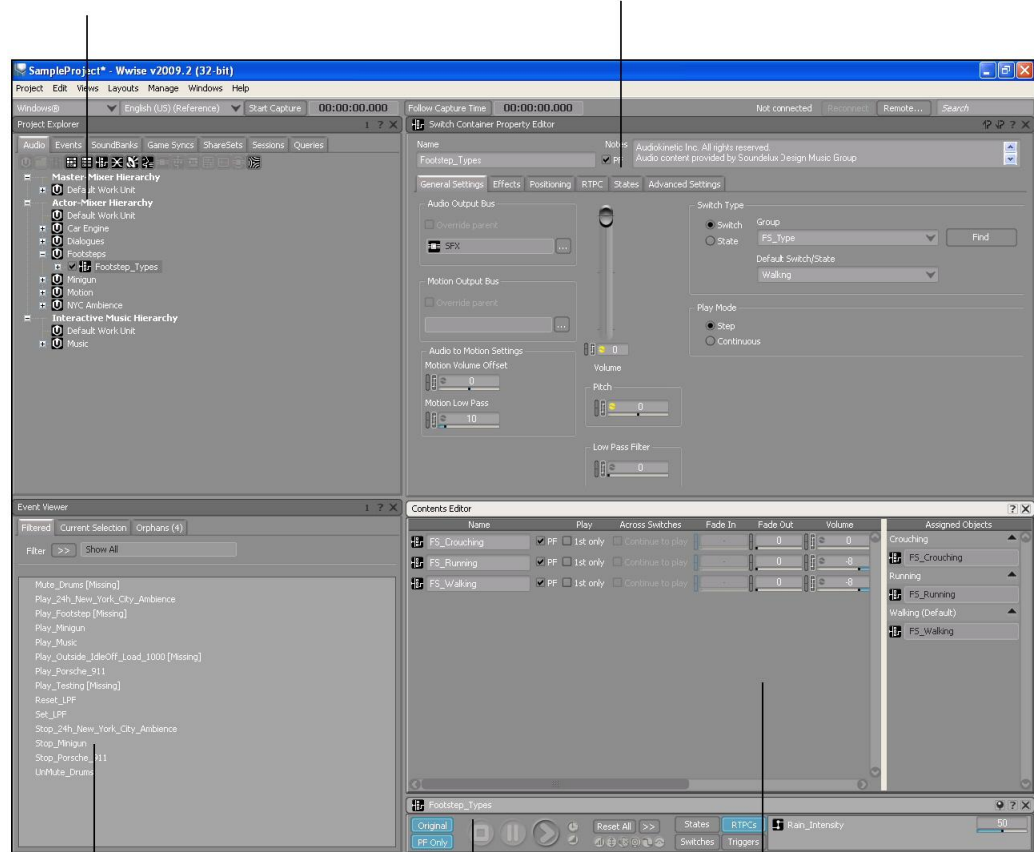
At the top of each layout is the menu bar and toolbar. The menu bar gives you access to all basic commands, such as saving projects, changing layouts, or opening views. The toolbar provides quick access to certain tools, such as the **Platform** or **Language Selectors**, the **Capture** tools, the **Remote Platform Connector**, and the **Search** tool.



When you first start Wwise, the **Designer** layout is displayed.

Project Explorer

Property Editor



Event Viewer

Transport Control

Contents Editor

It is comprised of the following views:

- **Project Explorer** - The main area where you manage and organize the various elements of your Wwise project. The **Project Explorer** contains the following tabs:
 - **Audio** - A hierarchical tree view, much like Windows Explorer and Mac Finder, where you can organize the sound, music, and motion assets in your project. The **Audio** tab has three main hierarchies: the **Master-Mixer Hierarchy**, the **Actor-Mixer Hierarchy**, and the **Interactive Music Hierarchy**.
 - **Events** - Displays the events, both action and dialogue events, in your project.
 - **SoundBanks** - Displays all the **SoundBanks** in your project.
 - **Game Syncs** - Displays all the switches, states, game parameters, and triggers in your project.
 - **ShareSets** - Displays all the effect and attenuation **ShareSets** in your project.
 - **Sessions** - Displays all the Soundcaster sessions in your project.

- **Queries** - Displays all the queries in your project.
- **Event Viewer** - Displays the different events that have been created for the current project. The **Event Viewer** has three different tabs: **Filtered**, **Current Selection**, and **Orphans**, each of which filters the events in a different way.
- **Property Editor** - Contains a collection of properties and behavior options that you can use to define the overall characteristics of a particular object within your sound, music, or bus structures.
- **Contents Editor** - Displays the object or objects that are contained within the parent object that has been loaded into the Property Editor. The Contents Editor also gives you quick access to some of the most common properties associated with each object, such as volume and pitch.
- **Transport Control** - Plays back your sound, music, and motion objects. The Transport Control contains the traditional controls associated with the playback of audio, such as play, stop, and pause.
- **Meter view** - Displays color-coded values per channel of three different kinds of levels, namely Peak, True Peak, and RMS.

Wwise Interface Basics

Standard operating system and Wwise-specific functionalities are described in the following sections.

- [Understanding Naming Conventions in Wwise](#)
- [Using Text Boxes](#)
- [Using Lists](#)
- [Using Sliders](#)
- [Using Tables](#)
- [Understanding the Visual Elements in Wwise](#)
- [Undoing and Redoing Actions in Wwise](#)

Understanding Naming Conventions in Wwise

When naming projects and objects in Wwise, you can use any Unicode-supported character.

However, to comply with the naming restrictions of certain game engines, some restrictions apply when naming the following elements in Wwise:

- SoundBanks
- Events
- Dialogue events
- Effect ShareSets
- Switch groups

- Switches
- State groups
- States
- RTPC game parameters
- Triggers
- Work units

For these elements, only numbers, unaccented Roman letters, and underscores are accepted. Additionally, the names cannot begin with a number.

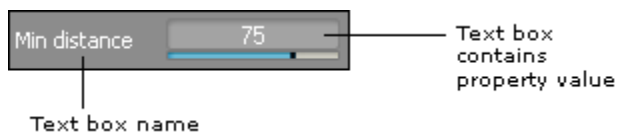


Note

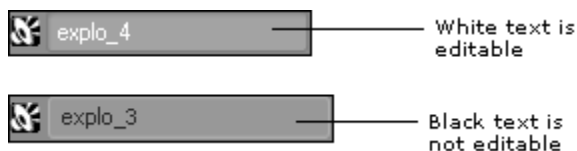
Most object names in Wwise are limited to 260 characters. SoundBank names, however, are given a 252-character limit by default; but, the actual limit is evaluated when trying to generate a SoundBank, which is written to disk within your project hierarchy's full file path. If the SoundBank's file path exceeds your operating system limit (for example, 260 characters on Windows 7 and 255 characters on Mac OS X), then generation will fail with *error 14: Can't write bank file*.

Using Text Boxes

Most of the views in Wwise contain fields or text boxes in which you can type property values or specific information about an object. The name of the text box identifies the type of information the field represents. Depending on the view, the name of the text box could be beside, above, or below it.



In different areas of the interface, text is displayed in a text box, but can't be edited. To help you distinguish between editable and non-editable text, Wwise displays the text using different colors. When the text appears in white, you can edit it. When the text appears in black, it can't be edited; but, you can copy it, if needed, by right-clicking the text and selecting *Copy Text*.



Most of the text boxes that contain property values also have a vertical bar or slider underneath that you can drag to change the value within these text boxes. For more information about sliders, refer to [Using Sliders](#).



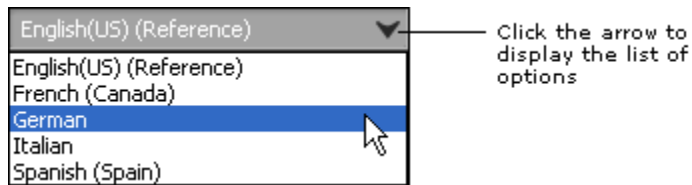
Note

The range of the slider may only be a subset of the possible values that can be entered for a property. To distinguish between the two ranges, we use “Default Slider Range” to refer to the limited range of the slider and “Input Range” to refer to the complete range of values that can be typed directly into the text box.

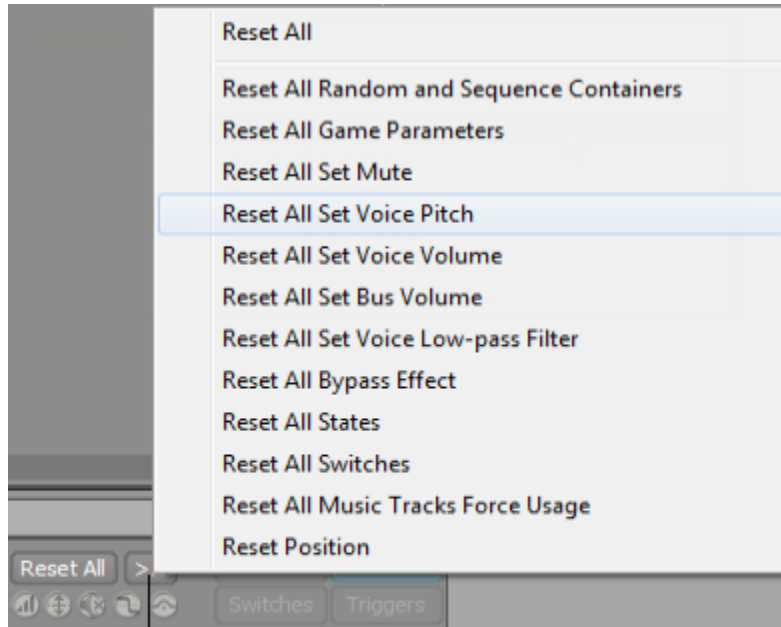
If you want to return a property value back to its default setting, Ctrl+click in the text box.

Using Lists

There are two types of lists in Wwise: a drop-down list and a shortcut list (sometimes called a contextual or context list). The drop-down list, referred to simply as a "list" in this help documentation, is a field that contains a series of pre-defined options. To display the list, click the arrow to the right of the field.

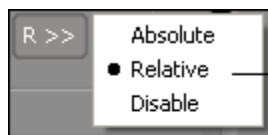


The Selector button (>>) displays a series of options or actions. The shortcut menu may or may not have a field associated with it to display the option selected. Click the button to display the menu options.

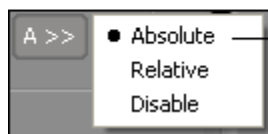


Click the Selector (>>) button to display the list of actions or options

When you select an option from a list, it will appear in the corresponding field beside it. If there is no field, the button itself will normally indicate which option you selected.



When the Relative option is selected, the letter "R" appears on the button.



When the Absolute option is selected, the letter "A" appears on the button.



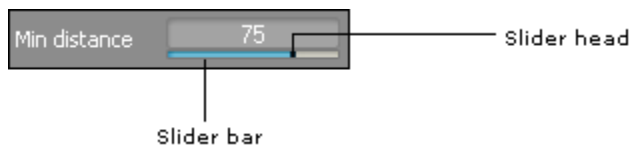
Note

If the list contains a series of actions, as is the case for the **Reset** button in the **Transport Control**, no indication will appear on the button.

Using Sliders

A slider is a control that sets a value from a continuous range of possible values. Most of the fields or text boxes that contain property values also have a horizontal slider right underneath them.

The horizontal sliders contain a slider head and a slider bar. The slider head is a little black point that represents the current property value. The slider bar is a blue bar that represents how much or how little the current value represents within the range of possible values.



The slider head will appear in different locations along the slider depending on where the default value falls within the range of possible values for each property. The slider bar will also start in different locations and move in different directions depending on where the default value falls within the range of possible values for each property.

You drag the slider head to the left or right to increase or decrease the property value. When you click and hold anywhere in the text box, the super slider appears, which makes it easier for you to fine-tune the property value. When you release the mouse button, the larger slider disappears. If you need to define the value more precisely, you can Shift+drag the slider to increase or decrease the value in smaller increments.

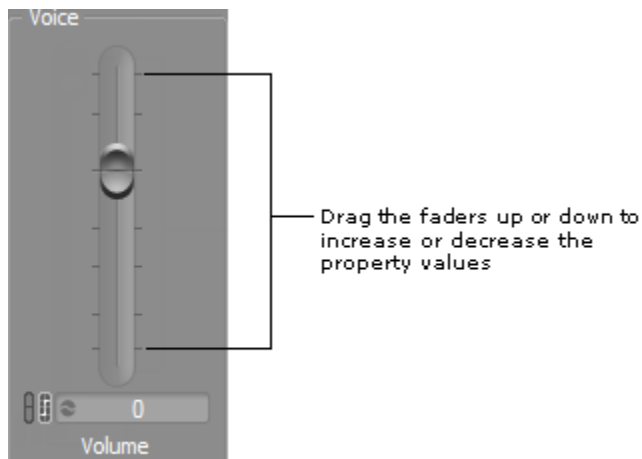


Note

Many of the sliders have a default range that is only a subset of the total range that can be input in the text box. To access a greater range of values for a particular property, you must enter a value outside the default slider range. To view the default slider and full input ranges for each property, refer to the Help.

Using Faders

Some properties, such as Volume, use vertical sliders or faders instead of horizontal sliders to change their values. The controls for Volume use vertical sliders to better simulate faders that are found on both hardware and software mixers. You can drag the fader up or down to increase or decrease these property values. If you need to define the value more precisely, you can Shift+click above or below the fader to increase or decrease the value in smaller increments.



Note

You can Ctrl+click anywhere on the slider or within a text box to return a property to its default value.

Using Tables

A table is an arrangement of data using a series of rows and columns. The information in the first column relates specifically to the information found in the other columns. In Wwise, tables are used to display information about Events, Presets, SoundBanks, and so on.

SoundBanks	Data Size	Max Size	Free Space	Type	Date Updated
<input checked="" type="checkbox"/> Level1	5 945 774	6000000	54 226	SFX	5/6/2008 3:24 PM
<input checked="" type="checkbox"/> Level2	131 682	500000	368 318	SFX, Voice	5/6/2008 3:24 PM
<input checked="" type="checkbox"/> Level3	5 945 774	7000000	1 054 226	SFX, Music, Motion	5/6/2008 3:24 PM

You can navigate through the items in the table by using the up and down arrow keys. In most cases, you can resize columns, sort information by each of the

different column headings, and edit the property values directly within the table.

To resize columns in a table:

1. Position the mouse pointer over a column divider.

The pointer changes to a double arrow.

2. Do one of the following:

Drag the divider to the right to increase the size of the column.

Drag the divider to the left to decrease the size of the column.

To sort information in a table:

1. Click a column title in the table.

The information in the table is sorted in ascending order according to the information in that column.

2. Click the column title again to sort the information in descending order.



Note

Where the order of items within a table plays an important role, you will not be able to sort by column title. Instead, you will be able to reorder the items manually by dragging them from one position to another.

To filter elements in a table:

1. Click the Search icon in the upper-right corner. Default Shortcut: Ctrl+F3.

A standard alphanumeric search field appears.



2. Enter any series of characters in the field, including numbers, letters, underscores or spaces.

Wwise filters from view all elements without a matching pattern of characters in any of their columns. With fewer elements listed, it's easier to find the one you need.



Note









Exceptionally, in the **File Manager** Wwise searches only the filenames of the **File** column.













Additionally, Wwise excludes objects hidden in a collapsed node in the **List View**, **Query Editor**, **MIDI Keymap Editor**, and **Reference View** from the search, thereby filtering them out by default unless a visible ancestor is kept.

- To close the search field and reset the filter, click the Close icon to the left of the Search icon or press Esc.

















Understanding the Visual Elements in Wwise


Wwise uses several icons or visual identifiers to help represent different elements in the interface. The following table describes each of the different visual elements used in Wwise.

Icon	Name	Represents
Sound Object Icons		
	Audio source	A separate abstraction layer between the audio file and the sound or motion object. It is linked to the audio file imported into your project and is where you define the conversion settings for the active game platforms.
	Plug-in source	An audio source that is created by a source plug-in coming from outside of Wwise.
	Sound SFX	A sound object containing sound effects.
	Sound Voice	A sound object containing voice over or character dialogue.
	Motion FX	An object containing motion data.
	Blend container	A group of one or more sounds, motion objects, and/or containers that are played back simultaneously. The sounds and containers within the blend container can be grouped into blend tracks where sound properties are mapped to game parameter values using RTPCs. Crossfades can also be applied between the sounds or motion objects within a blend track based on the value of a game parameter.
	Random container	A group of one or more sounds, motion objects, and/or containers that are played back in a random order.
	Sequence container	A group of one or more sounds, motion objects, and/or containers that are played back according to a specific order or playlist.

Icon	Name	Represents
	Switch container	A group of sounds, motion objects, and/or containers that are organized into a series of switches or states, so that they can be played when the associated switch or state is called by the game.
	Actor-Mixer	A hierarchical structure of one or more sounds, motion objects, containers, and/or actor-mixers. You can use an actor-mixer to apply properties to all objects below it.
	Audio Bus	A sound object grouping that allows you to work with different sound and music structures within a game. For example, you can group all the music structures under one audio bus and all the player sound structures under another.
	Auxiliary Bus	A sub-grouping of sound objects anywhere in the project for adjusting volume, channel configuration, positioning, and RTPC, as well as applying effects, states, or mixer plug-ins before routing back to a parent audio bus. Ducking, voice, and HDR mix adjustments are not possible in an auxiliary bus.
	Motion Bus	A grouping of many different motion structures within a game. For example, you can group all the explosion structures under one motion bus, all the vehicle structures under another motion bus, and so on.
	Work Unit	A distinct XML file that allows you to divide up your project into separate segments so that different members of your team can work on different parts of the project concurrently. These XML files can be easily managed by your source control system.
Music Object Icons		
	Music Track	A music object that contains arrangements of individual music clips that are displayed in waveform so that you can visually align them in a music segment.
	Random Music Track	A music track that plays back its sub-tracks in random order each time its parent segment is played.
	Sequence Music Track	A music track that plays back its sub-tracks in sequential order each time its parent segment is played.
	Music Segment	A music object that contains music tracks that can be aligned using sync points for musical arrangements in interactive music.
	Music Playlist Container	A group of one or more segments that are organized in a particular way so that they can be played back in a random order or according to a specific order.
	Music Switch Container	A group of one or more music segments and containers that are organized into series of

Icon	Name	Represents
		switches or states, so they can be played when the associated switch or state is called by the game.
Other Project Element Icons		
	Event	A method to trigger audio or motion in game using an action or series of actions, such as play, mute, and pause, that have been applied to one or more Wwise objects.
	Dialogue Event	A method to trigger audio or motion in game using a combination of state groups and states arranged into paths that have been assigned to an object in Wwise.
	SoundBank	A group of events, Wwise objects, and media that will be loaded into the game's platform memory at a particular point in a game.
	Switch Group	A collection of related switches that have been grouped together to help manage the different alternatives that exist for a given element within the game.
	Switch	An alternative that exists for a particular element within the game.
	State Group	A collection of related states that have been grouped together to help manage the global changes that occur in the game environment.
	State	A global offset or adjustment to the game audio properties that represent changes in the physical and environmental conditions in the game.
	Game Parameter (RTPC)	A parameter in your game, such as speed and RPMs in a car racing game, that can be mapped to Wwise property values using RTPCs.
	Trigger	A particular action in game that leads to the playback of one short piece of music.
	Effect ShareSet	Audio effect settings that can be used to enhance the audio in your game. These settings have been saved as a ShareSet and can be shared between objects.
	Attenuation ShareSet	Attenuation settings related to the volume of a sound based on its distance from the listener. These settings have been saved as a ShareSet and can be shared between objects.
	Conversion Settings ShareSet	Conversion settings, which include sample rate, audio format, and number of channels, help to define the overall quality of your audio output. These settings have been saved as a ShareSet and can be applied and shared between objects.
	Soundcaster Session	A group of sound, music, motion, and event modules arranged in a particular order that have been saved along with game sync settings for the purposes of a simulation.

Icon	Name	Represents
	Mixing Session	A group of busses and/or objects, along with their corresponding properties, that have been saved within a type of mixing console for the purposes of fine-tuning the audio mix of your game.
	Query	A specific set of search criteria used to find a particular object or project element.
Property Value Indicators		
	Link	A property setting that is linked to the settings of other active game platforms.
	Unlink	A unique property setting that is not linked to the settings of other active game platforms.
	Partial Unlink	The property setting for the current platform is linked, but one or more corresponding property settings of other active platforms are unlinked.
	RTPC - Disabled	This property value is not tied to an in-game parameter value.
	RTPC - Enabled	An in-game parameter value is tied to this property value. This means, for example, that the speed of a car in-game can be tied directly to the pitch property in Wwise. As the speed of the car increases in-game, the pitch in Wwise will increase in real time.
	Randomizer Enabled	A property value to which a Randomizer effect has been applied.
	Randomizer Disabled	A property value to which no Randomizer effect has been applied.
Property Shading		
	Unsupported feature	If a property is shaded in blue, it means that the feature is not supported on the current platform.
Preset Icons		
	Save Preset	A command to save the current state of all values within the current view to a preset.
	Load Preset	A command to load a previously saved preset.
View Icons		
	Help	A command to display the online help for that particular view, window, or dialog box.
	View Settings	A command to display a dialog box with a series of settings that can be used to define the view.
	Collapse View	A command to collapse the floating view.
	Expand View	A command to expand the floating view.

Icon	Name	Represents
	Close View	A command to close the floating view.

Undoing and Redoing Actions in Wwise

You can undo most actions that you perform in Wwise, such as changing a property value, moving an object, or creating an event. If you undo an action by mistake, you can redo the last action to return to the previous value or state.

To undo an action, click **Edit > Undo** Action Name or press **Ctrl+Z**. You can undo up to the last 200 actions.

To redo an action, click **Edit > Redo** Action Name or press **Ctrl+Y**. You can invoke one redo command for each undo action.

Personalizing Your Workspace

There are several ways to customize the Wwise work environment. The following table summarizes the types of customization that are available.

Area of Customization	Description
Working with Views	Defines the size and placement of flat and floating views.
Working with Layouts	Defines which layout is displayed, which views are displayed within the layout, and where they are displayed.
Setting User Preferences	Defines your particular user preferences.

Working with Views

The Wwise interface is made up of a series of views. A view is a separate window in the interface that contains a group of information or commands related to a specific task.

There are two different types of views: floating views and flat views. Flat views are the views that are docked into a layout, whereas floating views float above the layout like windows. Both types of views can be repositioned and resized.

To display a view:

1. From the menu bar, click **Views > View Name**.

The selected view is displayed as a floating view.

To switch between floating views:

1. From the menu bar, click **Windows > View Name**.







The selected view becomes active at the front of the stack of floating windows.

Related Topics

- [Syncing a Group of Views](#)
- [Understanding a View's Title Bar Icons](#)
- [Managing Floating Views Within a Layout](#)
- [Resizing the Views within a Layout](#)
- [Adding/Removing Views from a Layout](#)
- [Working with Layouts](#)

Understanding a View's Title Bar Icons

Views can have a series of icons on the right side of the title bar. Each icon performs a different command related to that view. Depending on the view, different combinations of these icons will be displayed. The following table describes each of the icons that can be found in the title bar of a view.

Icon	Name	Description
	Save Preset	Saves the current state of all values within the current view to a preset.
	Load Preset	Loads a previously saved preset.
	Sync Group	Syncs the view to a particular group (1-4). When a view is synced to a group, all views within the group will always display the same tab/selection. For example, let's say you have three different Project Explorers in three different layouts all synced to group 1. If you switch to the Game Syncs tab in one of the Project Explorers , all Project Explorer views within that group will automatically switch to the Game Syncs tab as well. This option is only available for the Project Explorer and Event Viewer views.
	Help	Displays the online help for that particular view, window, or dialog box.
	View Settings	Displays a dialog box with a series of settings that define what is displayed in the view.
	Close View	Closes the floating view.

Managing Floating Views Within a Layout

When working within a particular layout, you can have many different floating views open at any one time. You can manage these floating views efficiently using the Windows manager. Within the Windows Manager, you can easily activate a particular floating view, maximize, minimize, and restore views, move views, and close views.

To manage floating views within a layout:

1. From the menu bar, click **Windows > Windows**.

The Windows dialog box opens.

2. From the View Name list, select one or more of the views and then click one of the following:

Activate - To activate the selected floating view and bring it to the front of the stack of windows.

Minimize - To minimize the selected floating views.

Maximize - To maximize the selected floating views.

Restore - To restore the selected floating views that had previously been minimized or maximized to their original size and location.

Move To - To move the selected floating views to a new location within your monitor(s).

Close Windows - To close the selected floating views.

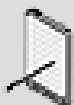
3. When you are finished, click **Close** to close the Windows dialog box.

Related Topics

- [Working with Views](#)
- [Resizing the Views within a Layout](#)
- [Adding/Removing Views from a Layout](#)
- [Restoring the Factory Layouts to Their Default Configuration](#)
- [Understanding a View's Title Bar Icons](#)

Syncing a Group of Views

Since you may have different instances of the **Project Explorer** and **Event Viewer** in several layouts, you may want these views to be synced. This means that any selection or movement made within the view is synced to the other views.



Note

Two views within the same layout can't belong to the same sync group.

To sync a group of views:

1. From the **Project Explorer** or **Event Viewer** title bar, click the **Sync Group** icon.

2. From the shortcut menu, select one of the following options:
 - No sync group** - To sync the view to no group.
 - Sync Group 1** - To sync the view to group one.
 - Sync Group 2** - To sync the view to group two.
 - Sync Group 3** - To sync the view to group three.
 - Sync Group 4** - To sync the view to group four.
3. To sync the Project Explorers and Event Viewers in other layouts, simply switch layouts and then repeat steps 1-2.

Related Topics

- [Working with Views](#)
- [Resizing the Views within a Layout](#)
- [Managing Floating Views Within a Layout](#)
- [Understanding a View's Title Bar Icons](#)
- [Adding/Removing Views from a Layout](#)
- [Docking/Undocking Views from a Layout](#)
- [Restoring the Factory Layouts to Their Default Configuration](#)

Working with Layouts

A group of views can be arranged together to create a layout. Wwise contains eight different default layouts that have been optimized to help you perform certain tasks or jobs. You can use different layouts depending on the job that you are doing.

The following layouts are available in Wwise:

- **Designer** - A special grouping of views that allows you to manage, build, and define the audio and motion assets in your game.
- **Profiler** - A special grouping of views that allows you to monitor and analyze the performance of game audio and motion elements as they occur.
- **SoundBank** - A special grouping of views that allows you to create, manage, and generate SoundBanks.
- **Mixer** - A special grouping of views that allows you to create prototypes and mix the various objects in Wwise.
- **Schematic** - A special grouping of views that allows you to view the hierarchy of objects and busses as a graphical representation. You can also edit properties and play back objects.
- **Interactive Music** - A special grouping of views that allows you to manage, build, and define the music assets involved in the interactive music portion of your game.

- **Dynamic Dialogue** - A special grouping of views that allows you to manage and build the dialogue events that drive the dynamic dialogue in your game.
- **Game Object Profiler** - A special grouping of views that allows you to monitor and analyze the performance of audio and motion from the game object's standpoint.

To switch between layouts:

1. From the menu bar, click **Layouts** > Layout Name.

The new layout is displayed.



Tip

You can also switch between layouts using the F5-F12 shortcut keys.

Related Topics

- [Working with Views](#)
- [Resizing the Views within a Layout](#)
- [Adding/Removing Views from a Layout](#)
- [Docking/Undocking Views from a Layout](#)
- [Restoring the Factory Layouts to Their Default Configuration](#)

Resizing the Views within a Layout

You can modify the default layouts by resizing the existing views within a layout.

To resize a flat view:

1. Position the mouse pointer over the splitter bar between two or more views.

The splitter bar becomes highlighted and the mouse pointer becomes a two-headed arrow.

2. Drag the splitter bar to resize all views that reside along the highlighted splitter bar.



Note

You can resize floating views as well. They are resized in the same way as other floating windows in Windows XP or Vista.

Related Topics

- [Adding/Removing Views from a Layout](#)
- [Docking/Undocking Views from a Layout](#)
- [Restoring the Factory Layouts to Their Default Configuration](#)
- [Working with Layouts](#)
- [Working with Views](#)

Adding/Removing Views from a Layout

Although the factory layouts in Wwise have been optimized to make the workflow as efficient as possible, there may be cases when you want to add or remove a view from a layout. A view, whether it be floating or flat, will be part of a layout until it is removed.

To add a new view to a layout:

1. From the menu bar, select **Views > View Name**.

The selected view appears as a floating view within the layout.

2. You can resize and move the floating view anywhere within your monitor(s).



Note

This floating view will always be displayed with the current layout until you close it.

To remove a view from a layout:

1. In the title bar of a floating or flat view, click the **Close icon (X)** to close the floating view.

If the view is docked within the layout, Wwise prompts you to confirm the layout modification.



Note

If you don't want Wwise to confirm every modification you make to a layout, you can clear the selected option “Warn when modifying docked layout” in the Layouts menu.

2. Click **Yes**.

The docked view is removed from the layout and the remaining views are resized to fill the empty space.

Related Topics

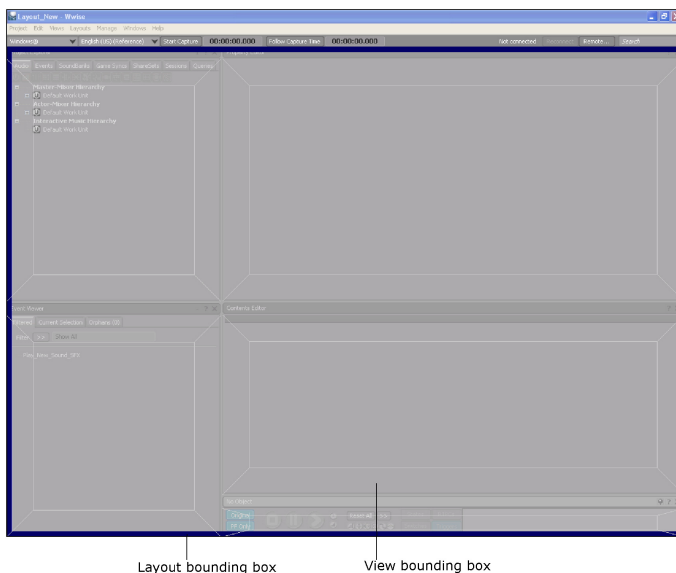
- [Docking/Undocking Views from a Layout](#)
- [Resizing the Views within a Layout](#)
- [Restoring the Factory Layouts to Their Default Configuration](#)
- [Working with Layouts](#)
- [Working with Views](#)

Docking/Undocking Views from a Layout

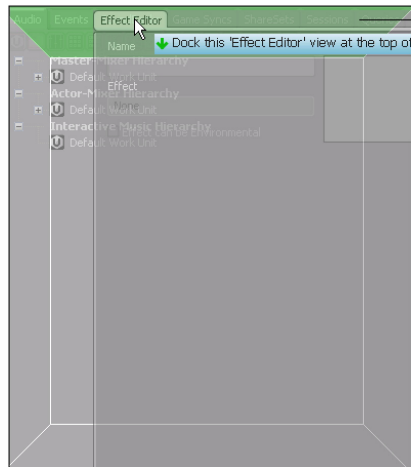
Although the default layouts have been designed to make the workflow as efficient as possible, there may be circumstances where you want to edit the contents and positioning of the views within a layout.

When docking a view to a layout, you have several choices as to where the new view can be docked. When Wwise enters “edit layout” mode, the layout is divided into different areas according to the various splitter bars that separate the views within the layout. Separate bounding boxes are also created around the following:

- The entire layout (represented by a thick blue line).
- Each view within the layout.

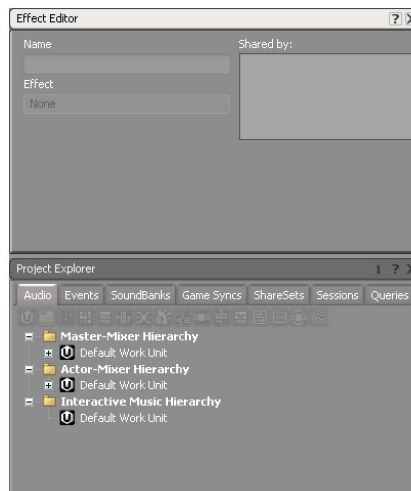


The four sides of each bounding box represent a different position where the new view can be docked within the layout. For example, let's say you want to add the **Effect Editor** to the **Designer** layout right above the **Project Explorer**. You can open the **Effect Editor** and then drag it to the top section of the bounding box for the **Project Explorer**.



Area of bounding box where the new view will be docked is highlighted in green.

This will split the area defined by the **Project Explorer** in two, placing the **Effect Editor** in the top half and the **Project Explorer** in the bottom half.



Effect Editor

Project Explorer

If you had wanted the Effect Editor to take up the entire width of the layout, you would have used the top layout bounding box instead of the view one.

To dock a floating view within a layout:

1. Click the view's title bar within the floating window and start dragging the view.

The interface enters “edit layout” mode.

2. Drag the floating view over the area of the particular bounding box where you want the view to be docked within the layout.

The area of the bounding box becomes highlighted in green.

3. Release the mouse button.

Wwise prompts you to confirm the layout modification.



Note

If you don't want Wwise to confirm every modification you make to a layout, you can clear the selected option “Warn when modifying docked layout” in the Layouts menu.

4. Click Yes.

The floating view becomes a flat view within the layout in the location specified.

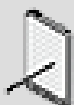
5. You can now resize the flat views to get the exact layout you want.

To undock a view from a layout:

1. Click the title bar of the view you want to undock from the layout and then start dragging the view.

Wwise enters “edit layout” mode.

2. Drag the view anywhere within the layout making sure that no bounding boxes are highlighted in green.

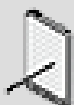


Note

If an area of a bounding box is highlighted in green, the view will simply move to a new location in the layout.

3. Release the mouse button.

Wwise prompts you to confirm the layout modification.



Note

If you don't want Wwise to confirm every modification you make to a layout, you can clear the selected option “Warn when modifying docked layout” in the Layouts menu.

4. Click Yes.

The flat view becomes a floating view.

Related Topics

- [Resizing the Views within a Layout](#)
- [Adding/Removing Views from a Layout](#)

- [Restoring the Factory Layouts to Their Default Configuration](#)
- [Working with Layouts](#)
- [Working with Views](#)

Restoring the Factory Layouts to Their Default Configuration

After editing a layout, you may decide that you want to return to the default layout configuration. Wwise allows you to restore the default layouts at any time. Be aware, however, that there is currently no way to save an edited layout, so when you restore a factory layout, you will lose the changes you made to the layout.

To restore the factory layouts to their default configuration:

1. From the menu bar, click **Layouts > Reset Factory Layouts**.

The Reset Factory Layouts dialog box opens.

2. Select the layouts that you want to restore to their default configuration.
3. Click **OK**.

The selected factory layouts are restored to their default configurations.

Related Topics

- [Resizing the Views within a Layout](#)
- [Adding/Removing Views from a Layout](#)
- [Working with Layouts](#)
- [Working with Views](#)

Setting User Preferences

You can set the user preferences to customize the following four aspects of Wwise:

- [Enabling Confirmation Messages](#)
- [Selecting External Audio Editors](#)
- [Setting the Output Buffer Latency](#)
- [Setting the Audio Channel Configuration](#)
- [Setting the Music Track Look-ahead Time](#)
- [Setting the Privacy Settings for the Project Launcher Web Browser](#)
- [Setting the Documentation Preferences](#)

Enabling Confirmation Messages

Confirmation messages can provide useful information when you are performing certain tasks in Wwise. While working with Wwise, you can disable

these messages one at a time by selecting the “Don't ask again” option in certain confirmation dialog boxes. If you want all these messages to appear again, re-enable them in the user preferences.

To enable confirmation messages:

1. From the menu bar, click **Project > User Preferences**.

The User Preferences dialog box opens.

2. In the Confirmation Messages group box, click **Reset**.

A confirmation dialog box appears.

3. Click **OK**.

From now on, confirmation messages will appear before certain actions are carried out in Wwise.

Selecting External Audio Editors

In Wwise, you can edit audio files directly in any audio file editor of your choice. First, however, you need to add each program to the list of available editors.

To select external audio editing software:

1. From the menu bar, click **Project > User Preferences**.

The User Preferences dialog box opens.

2. In the External Editors group box, click **Add...**

The Open File dialog box opens.

3. Select the audio editor executable (EXE) and click **Open**.

The editor is added to the list of available audio editors.



Note

To delete an editor from the list, select it and click **Delete**.

4. Click **OK** to save your settings.

From now on, the editor you have added will be available directly from Wwise.

To select the default external audio editor:

1. In the list of editors, select the one you wish to make default.

2. Click *Set As Default*



Tip

In most views in Wwise, press Ctrl-E to open the selected items in the default External Editor.

Setting the Output Buffer Latency

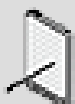
When you are playing sounds in Wwise, the Wwise sound engine uses pre-filled buffers to reduce playback latency. By default, the sound engine uses two of these buffers. However, you can adjust the number of buffers the sound engine uses to correct certain performance issues you might encounter while using Wwise.



Note

You cannot change the number of buffers being used while sounds are being played back, or when you are connected to a game.

If you notice voice starvation happening when you play back sound objects in Wwise, try increasing the number of output buffers. If, on the other hand, you would like to decrease playback latency, try decreasing the number of buffers.



Note

Changing the number of output buffers resets the sound engine. As a consequence, any changes made to sound object property values through events are lost. For example, if you were auditioning a sound which had its volume reduced through an event, then changed the number of output buffers, the sound would return to its original volume.

To set the output buffer latency:

1. From the menu bar, click **Project > User Preferences**.

The User Preferences dialog box opens.

2. In the Sound Engine group box, select a number of output buffers from the list. The number in parentheses represents the corresponding latency.
3. Click **OK** to save your settings.

From now on, the sound engine will use the number of output buffers you have specified when playing sounds in Wwise.

Setting the Audio Channel Configuration

System Default Channel Configuration

- From the menu bar, click **Audio > System Default Channel Configuration**
- By default, Wwise use the speaker setup configuration from the Windows control panel. Select this option to choose value selected in the Windows control panel.

Stereo Channel Configuration (Speakers)

- From the menu bar, click **Audio > Stereo Channel Configuration (Speakers)**
- For more information on panning rules (speakers, headphones), refer to [Speakers vs Headphones Panning Rules](#).

Stereo Channel Configuration (Headphones)

- From the menu bar, click **Audio > Stereo Channel Configuration (Headphones)**
- For more information on panning rules (speakers, headphones), refer to [Speakers vs Headphones Panning Rules](#).

5.1 Channel Configuration

- From the menu bar, click **Audio > 5.1 Channel Configuration**

7.1 Channel Configuration

- From the menu bar, click **Audio > 7.1 Channel Configuration**



Note

You can select **5.1 or 7.1 Channels Configuration** while Windows control panel is set to stereo. Be aware that this might force DirectSound to downmix from 5.1 or 7.1 to stereo.

Setting the Music Track Look-ahead Time

Audio playback is always streamed while authoring in Wwise. If you have specified streaming and defined music track look-ahead time in the **Music Track Property Editor**, these values will be used during streaming. If you have

not selected the streaming option, the default music track look-ahead time of 200 milliseconds will be used. To increase flexibility and to avoid sync and voice starvation problems during playback, you can adjust the music track look-ahead time as a user preference.

To set the music track look-ahead time:

1. From the menu bar, click **Project > User Preferences**.

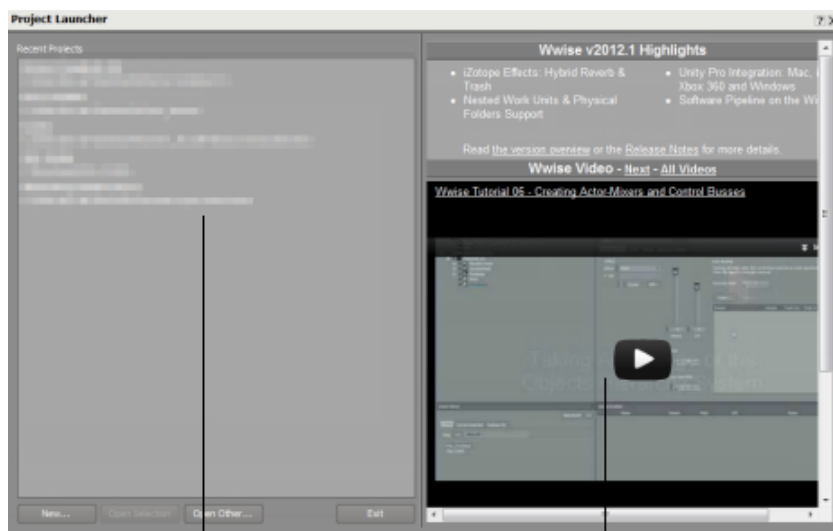
The **User Preferences** dialog box opens.

2. In the **Sound Engine** group box, type the amount of look-ahead time in milliseconds in the **Music Track look-ahead time (ms)** text box.
3. Click **OK** to save your settings.

From now on, the sound engine will use this amount of look-ahead time to seek streamed data when playing music in Wwise for music tracks that do not have defined streaming settings.

Setting the Privacy Settings for the Project Launcher Web Browser

The **Project Launcher** dialog box can be configured to include a web browser. This web browser will display links to various learning materials. If you decide to display the web browser within the Project Launcher, you can also decide whether you want to send information about the version of Wwise you are currently using along with information about the version of Windows upon which Wwise is currently running. The information collected will be used to tailor the information that is displayed in the web browser or for statistical purposes.



List of recently opened projects

Web browser displays links to various learning materials.

To set the privacy settings for the Project Launcher web browser:

1. From the menu bar, click **Project > User Preferences**.

The **User Preferences** dialog box opens.

2. In the **Project Launcher - Privacy** group box, select the **Show Web Browser** option to display the web browser in the **Project Launcher** dialog box.

The web browser displays the latest Wwise news as well as links to various learning materials.

3. If the web browser is enabled, information can be sent to Audiokinetic about Wwise and the system upon which Wwise is running. To send this information, select the following two options:

Send Wwise version information - Sends information about the version of Wwise currently running. The information sent includes the following:

- Wwise version
- Wwise build number
- Wwise architecture: 32 or 64 bits
- Wwise patch
- **Send system and Windows information** - Sends information about the version of Windows upon which Wwise is currently running. The information sent includes the following:
 - Windows version
 - Windows service pack



Note

This information will be used to tailor the information that is displayed in the web browser. For example, only information related to the version you are running will be displayed. The information collected will also be used for statistical purposes.

4. Click **OK** to save your settings.

Setting the Documentation Preferences

The Wwise Help can be set to open a local CHM file or to open the online version of the help documentation available at audiokinetic.com. Both source types are available in English and Japanese.

To set the Wwise Documentation preferences:

1. From the menu bar, click **Project > User Preferences**.

The **User Preferences** dialog box opens.

2. In the **Documentation** group box, select **CHM** from the **Source** options to use the locally stored Wwise compiled help file for contextual help, or select **audiokinetic.com** to use the Audiokinetic online web documentation for contextual help.
3. Select either **English** or **Japanese** from the **Language** options.

The selected language will be used in the Wwise Help (including the contextual help), Wwise User's Guide, and the Wwise Fundamentals help document options.

4. Click **OK** to save your settings.

Speeding Up the Way You Work

As you become more familiar with Wwise, you can start using some of the more advanced features. The following tools will increase your productivity by giving you quick access to certain operations and commands:

- [Using Keyboard Shortcuts](#)
- [Using Shortcut Menus](#)
- [Using the Batch Rename](#)

Using Keyboard Shortcuts

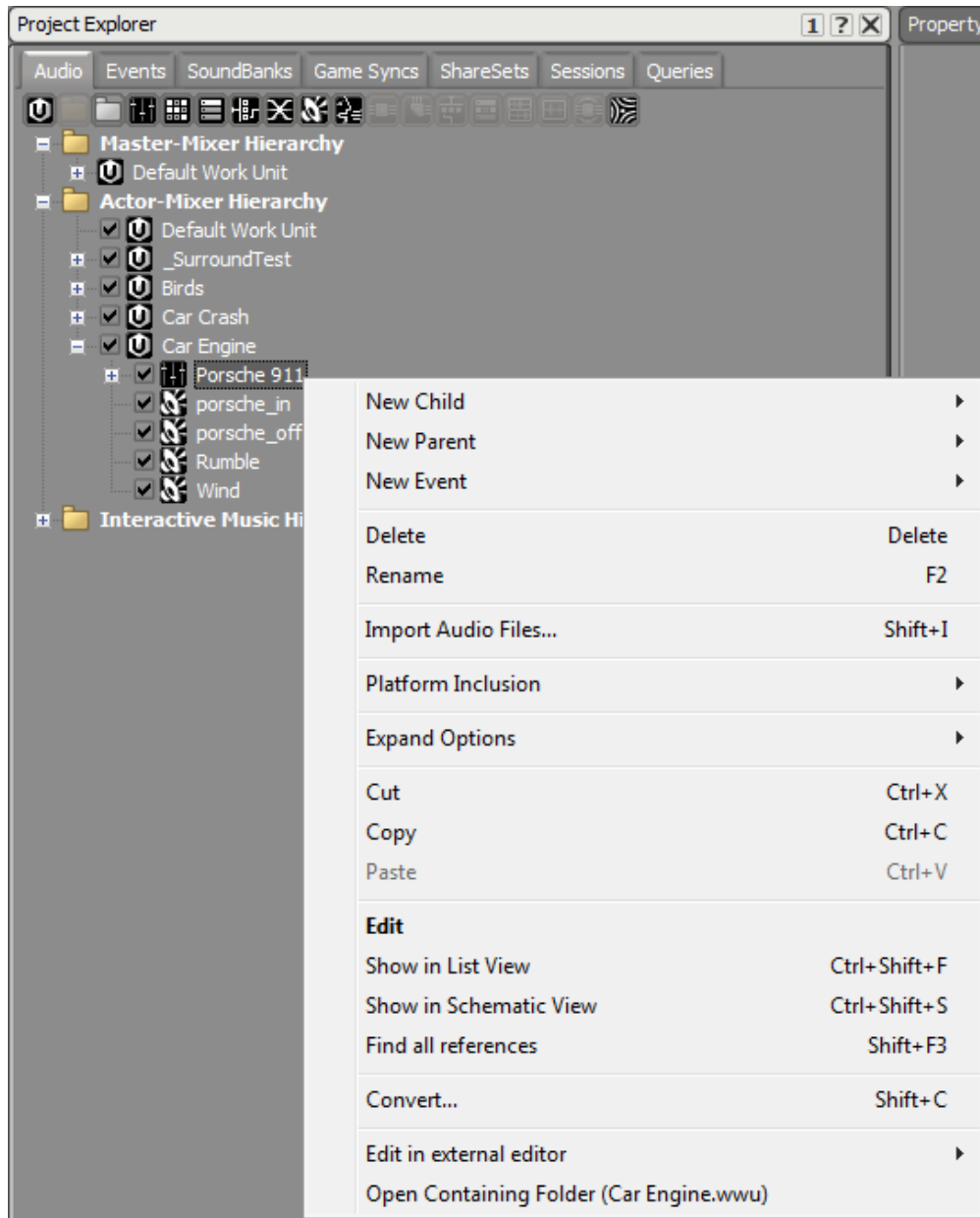
Many of the commands or operations within Wwise have been mapped to a key or key combination on your keyboard. You can use the keyboard shortcut instead of the mouse to perform any of these actions or commands. For example, you can press **Ctrl+S** to save your project instead of using your mouse and clicking **File > Save**.

For a complete list of keyboard shortcuts, refer to [Appendix B, Shortcuts](#).

Using Shortcut Menus

Shortcut menus are a list of commands that relate to an object, project element, or specific area on the Wwise interface. You can access these menus by right-clicking an object or specific area of the interface.

For example, when you right-click an object in the **Project Explorer**, a shortcut menu appears with several commands. You can use these commands to cut, copy, or paste an object, create a parent or child object, create events associated with a specific object, convert a sound or music object, and so on.



Although these menus are contextual, there are several commands that appear across menus. Along with the standard Windows commands, such as Cut, Copy, Paste, Delete, and Rename, you can access many of the following Wwise-specific commands by right-clicking different objects or areas of the interface:

- **Edit** - To load the selected object or project element into its respective editor.
- **Find in Project Explorer** - To highlight the selected object or project element in the **Project Explorer**. When using this command, you must also specify the sync group, if any, to which the **Project Explorer** belongs.
- **Details** - To display the location of the object or project element in the Project hierarchy.

- **Find All References** - To display a list of project elements that contain direct references to the current object. The list of direct references is displayed in the Reference view.
- **Show in Schematic View** - To display the selected object in the Schematic view.
- **Edit in external editor** - To open a list of external editors, if defined in your User Preferences.
- **Open containing folder** - To prompt Windows Explorer (or Mac Finder) to the location of the object's parent element.

Using the Batch Rename

The **Batch Rename** view allows users to rename editable objects or to change their associated notes in one step with a powerful simultaneous replace, remove, and insert mechanism. This eliminates the need for long, tedious, or potentially error-prone tasks of individually renaming objects.

Descriptions and examples of using the **Batch Rename** view are provided in the following pages.

- [Opening the Batch Rename View](#)
- [Specifying Batch Rename Settings](#)
- [Applying Batch Rename Changes](#)

Opening the Batch Rename View

The **Batch Rename** view can be prompted from several locations with or without objects to rename loaded in its view.

To prompt the Batch Rename view without objects:

1. Select **Views > Batch Rename** or use the shortcut. Default shortcut: Ctrl + F2.

The **Batch Rename** view opens with an empty **Preview** panel.

To prompt the Batch Rename populated with selected objects:

1. Select editable objects in the **Project Explorer** (or other object view like the **List View**) you want to run rename operations on.
2. Select **Batch Rename...** from the shortcut (right-click) menu or use the keyboard shortcut.

The **Batch Rename** view opens with its **Preview** panel populated with the selected objects.

To add objects to the Batch Rename view:

1. With the **Batch Rename** open, select editable objects in the **Project Explorer** (or other object view) you want to run rename operations on.
2. Select **Batch Rename...** from the shortcut menu or using the keyboard shortcut, or drag the selected objects over to the **Preview** panel.

The selected objects appear in the **Preview** panel, replacing previously selected objects.



Note

Hold Shift while dragging objects over from the **Project Explorer** (or other object view) to the **Preview** panel to add to, not replace, the objects currently listed in the panel.

Related Topics

- [Specifying Batch Rename Settings](#)
- [Applying Batch Rename Changes](#)

Specifying Batch Rename Settings

After [Opening the Batch Rename View](#), actions taken in the **Settings** panel of the **Batch Rename** view define the changes that will apply to the objects listed in the **Preview** panel.

The **Settings** panel consists of three sections:

- **Replace** - See more in [Replacing Content](#).
- **Remove** - See more in [Removing Content](#).
- **Insert** - See more in [Inserting Content](#).

Replacing Content

The **Replace** section is the first of three sections in the **Batch Rename Settings** panel. In this section, users can replace text in object names or notes using very simple to very complex replacement patterns.

To replace content:

1. Specify in the **Apply To** list if the rename operation should apply to the listed objects' names or comments.
2. Select **Replace**.

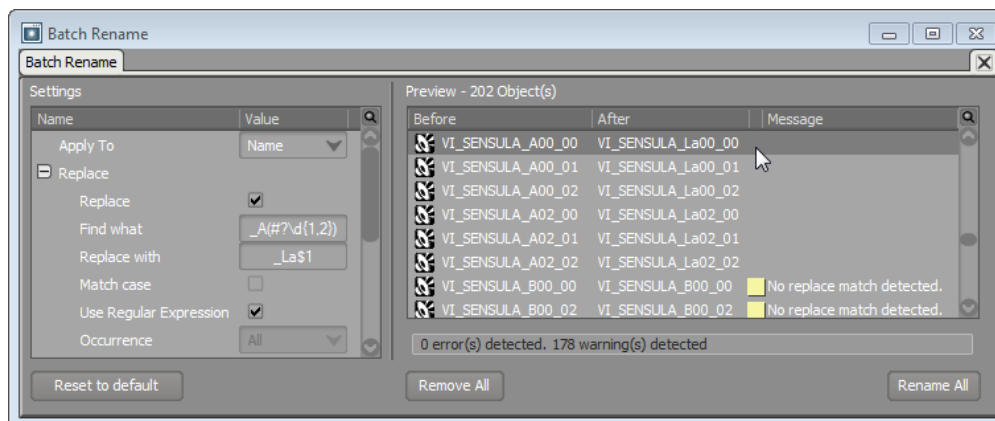
3. Enter the text to find, optionally in the form of a regular expression pattern, in the **Find what** field.
4. Enter the replacement text, optionally in the form of a regular expression back reference, in the **Replace with** field.
5. Select **Match case** if you want to set the **Find what** entry to only find text with the same letter case (upper or lower).



Note

Match case is not applicable to regular expression patterns, which can be set as needed to recognize case.

6. Select **Use Regular Expression** if you want Wwise to interpret the **Find what** and **Replace with** entries as regular expression syntax.
7. Specify in the **Occurrence** list what to do if there is more than one matching occurrence of the specified pattern:
 - **First** - Include only the first matching occurrence.
 - **Last** - Include only the last matching occurrence.
 - **All** - Include every match of the pattern in the object name or comment.
8. Review your changes then click **Rename All** to replace the content.



Example: Replacing with regular expressions

Let's take the Wwise Sample Project as an example. In its **Actor-Mixer Hierarchy**, the **MIDI Work Unit** has dozens of MIDI objects that are, in part, named by their corresponding absolute note: C00, D00, E00, F00, G00, A01, B01, C02, and so on. But, imagine your project is transferred to your French studio where they use fixed-doh solfège notes: Do, Re, Mi, Fa, Sol, La, Si, Do, and so on. The French studio audio designers will appreciate it if we rename all these objects. Here is how we'd do it:

1. Press Ctrl and click on the **MIDI Work Unit**.

The **MIDI Work Unit** hierarchy expands, revealing its many objects.

2. Select all the Kalimba, Sansula, Voice, and Wood Sansula objects in the **Project Explorer**. Then open the shortcut menu and select **Batch Rename...**

The **Batch Rename** view appears with the selected objects in its **Preview** panel.

3. In the **Settings** panel, if not already done, set **Apply To** to **Name**.

The **Preview** panel lists the current object names under the **Before** column.

4. Enable **Replace** and, a few rows down, enable **Use Regular Expression**.

The **Find what** and **Replace with** fields activate, ready to accept and interpret entries as regular expressions.

5. Enter `_C(#\d{1,2})` in the **Find what** field. Then click the **Preview** panel to see the forecast results.

All objects with names containing `_C`, with or without `#`, and followed by one or two digits are listed in the **After** column without these matching characters, prompting an error message, "Resulting name is already used by a sibling object or is otherwise reserved", for a couple of these objects. All other objects list a "No replace match detected" message.

6. Enter `_Do$1` in the **Replace with** field. Then click the **Preview** panel to see the forecast results. Instead of just removing the matching `C` note, the **After** column now lists `Do` in the place of the `C`.
7. Click **Rename All** to apply the change.

For each matching object, names update and a "Successfully renamed" message displays in the **Preview** panel.

We would then repeat the operation for the remaining six notes.

The following table gives a more detailed explanation of how our regular expression, `_C(#\d{1,2})`, with its replacement, `_Do$1`, worked.

Before	After	Explanation
VI_VOICE_33_C#6	VI_VOICE_33_Do#6	<ul style="list-style-type: none"> • <code>_C</code> literally matches <code>_C</code> in the name, just like a non-regular expression replacement would. • <code>#?</code>, zero or one match of <code>#</code>, matches the one instance of <code>#</code>. • <code>\d{1,2}</code>, one or two digits, matches the 6. • The parentheses around <code>#?\d{1,2}</code> put the matching <code>#</code> and 6 in a group. <p>In the replacement:</p>

Before	After	Explanation
		<ul style="list-style-type: none"> The match is replaced by <code>_Do</code> and <code>\$1</code>, the first group, whose value is <code>#6</code>.
VI_WOOD_SANSULA_C00_HI	VI_WOOD_SANSULA_Do00_HI	<ul style="list-style-type: none"> <code>_C</code> literally matches <code>_C</code> in the name, just like a non-regular expression replacement would. <code>#?</code>, zero or one match of <code>#</code>, matches zero instances of <code>#</code>. <code>\d{1,2}</code>, one or two digits, matches the <code>00</code>. The parentheses around <code>#?\d{1,2}</code> put the matching <code>00</code> in a group. <p>In the replacement:</p> <ul style="list-style-type: none"> The match is replaced by <code>_Do</code> and <code>\$1</code>, the first group, whose value is <code>00</code>.

Related Topics

- [Opening the Batch Rename View](#)
- [Specifying Batch Rename Settings](#)
- [Applying Batch Rename Changes](#)

Removing Content

The **Remove** section is the second of three sections in the **Batch Rename Settings** panel. In this section, users can remove text in object names or notes using a simple character positioning system.

To remove content:

1. Specify in the **Apply To** list if the rename operation should apply to the listed objects' names or comments.

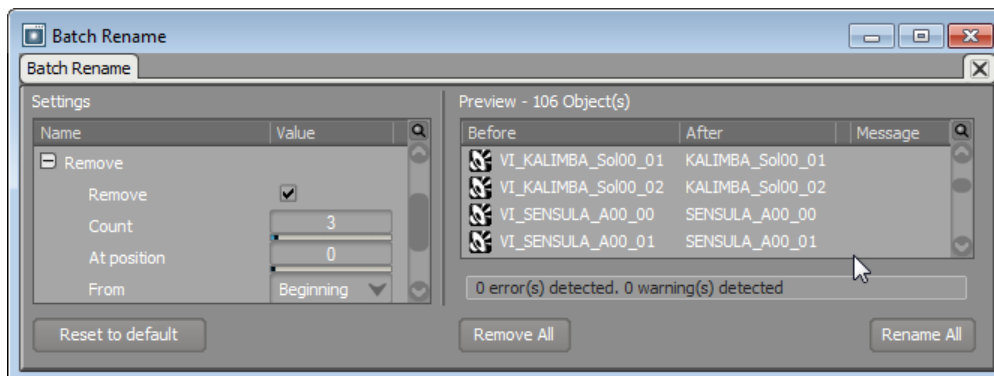
The **Before** and **After** columns in the Preview panel update with the selected object content.

2. Enable **Remove**.

Count, **At position**, and **From** activate.

3. Specify the number of characters to remove by entering a whole number in the **Count** field or moving its slider to the desired value.
4. Specify the character position at which the remove operation should be performed. To do so, enter a whole number in the **At position** field or move the slider to the desired value.
5. Select the direction from which the **At position** should be counted:

- **Beginning** (of the name or comment)
 - **End** (of the name or comment)
6. Review your changes then click **Rename All** to remove the content.



Example: Removing from your matches

Taking the Wwise Sample Project as an example, suppose the director of the Sample Project decides that we should shorten the project's object names by getting rid of extraneous information. In this case, all the SFX under the Kalimba, Sansula, Voice, and Wood Sansula blend containers are prefixed with "VI_", as in Virtual Instruments. It is not key for defining these objects, so we decide to remove it.

1. Open the **Batch Rename** view and, from the **Project Explorer**, drag all the SFX of those blend containers into the **Preview** panel.

One hundred and six objects beginning with "VI_" are listed. (If not, check that the **Apply To** list is set to apply this batch rename action to the objects' **Name**, not their **Notes**.)

2. Moving to the **Settings**' second hierarchy, enable **Remove**.
3. Enter 3 in **Count**.
4. Specify 0 as the **At position**.
5. Select **Beginning** in the **From** list.

This covers the names' first three characters, which are "VI_". (If you only wanted to remove that underscore between the "VI" and the instrument name, it would be a **Count** of 1 with an **At position** of 2.)

6. Click the **Preview** panel.

The **After** column updates to show all the object names without "VI_", giving names such as *SENSUSLA_A00_00*.

7. Click **Rename All** to apply the change.

For every object in the **Batch Rename**, names update and a "Successfully renamed" message displays in the **Preview** panel.

Related Topics

- [Opening the Batch Rename View](#)
- [Specifying Batch Rename Settings](#)
- [Applying Batch Rename Changes](#)

Inserting Content

The **Insert** section is the third of three sections in the **Batch Rename Settings** panel. In this section, users can insert text and printf format number patterns in object names or notes using simple replacement and character positioning.

To insert content:

1. Specify in the **Apply To** list if the rename operation should apply to the listed objects' names or comments.
2. Enable **Insert**.
3. Specify the type of insertion it will be in **Insert what**:

- **Text** - Just the literal text entry of **To insert**.

start at becomes inactive.

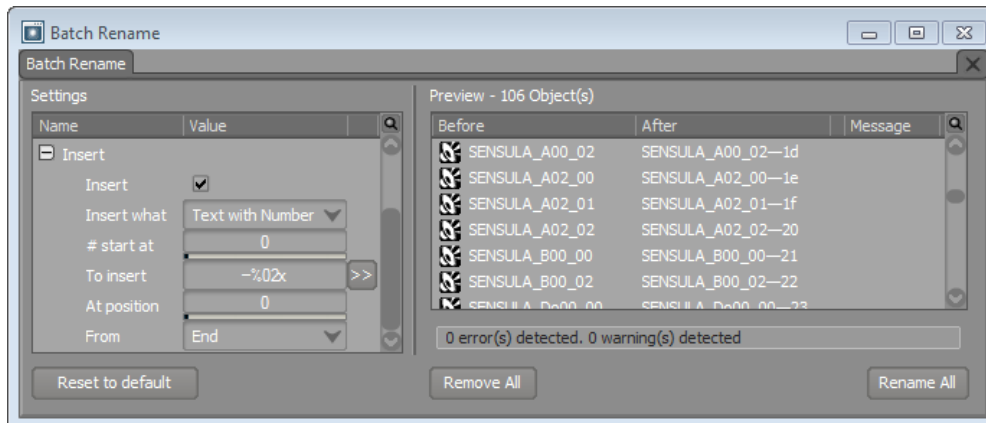
- **Text with Number** - The text entry of **To insert** with recognition of C++ printf format number patterns.

start at becomes active.

4. If you chose **Text with Number**, then specify a number at which the count should begin for your entered C++ printf format number pattern by entering a whole number in the # **start at** field or moving the slider to the desired value.

If you chose **Text**, this field should be inactive.

5. Specify the insert text in the **To insert** field. Keep in mind that if you specified **Text with Number**, then Wwise expects an appropriate C++ printf format number pattern here.
 - Optionally, use the selector to enter predefined text in the form of commonly used C++ printf format number patterns.
6. Specify the character position at which the insert operation should be performed. To do so, enter a whole number in the **At position** field or move its slider to the desired value.
7. Select the direction from which the **At position** should be counted:
 - **Beginning** (of the name or comment)
 - **End** (of the name or comment)
8. Review your changes then click **Rename All** to insert the content.



Example: Inserting numbers

Let's take the Wwise Sample Project as an example again. Suppose we want to track all the Virtual Instrument SFX taken from the Sample Project's Kalimba, Sansula, Voice, and Wood Sansula blend containers of the Actor-Mixer Hierarchy's MIDI work unit. As different virtual instruments, it is appropriate they have their own containers. In our [Example: Removing from your matches](#), we already removed a group identifying prefix, "VI_", from these virtual instrument SFX objects because it was not key to identifying them. The QA team has decided it would be easier to track their names if they began with a unique identifying number, so that's what we'll do: insert a unique number in front of these SFX objects.

1. Select all those containers' SFX objects in the **Project Explorer** and click our **Batch Rename** shortcut: Ctrl+F2 (by default).

The **Batch Rename** view appears with the selected SFX objects in its **Preview** panel.

2. Assuming **Name** is the selected **Apply to** option, enable **Insert** in the **Settings** panel.

The insert fields (**Insert**, **Insert what**, **To insert**, **At position**, and **From**) activate.

3. Choose **Text with Number**.

The **# start at** field activates.

4. Leave the **#start at** as 0.
5. In the **To Insert** field, enter an em-dash followed by a lower-case hexadecimal printf symbol with zero padding to ensure it is at least two digits: `—%02x`.
6. Specify 0 as the **At position**.
7. Select **Beginning** in the **From** list.
8. Click the **Preview** panel.

9. The **After** column updates to show all the object names suffixed with an em-dash (providing a nice separation from the original object name) and a two digit hexadecimal number, such as *SENSULA_A02_01—1f*.
10. Click **Rename All** to apply the change.

For every object in the **Batch Rename**, names update and a "Successfully renamed" message displays in the **Preview** panel.



Note

The **Batch Rename** view inserts numbers according to the alphabetical order in which objects are found in the **Preview** panel before any rename applications. So, objects A, B, and D will respectively be numbered A0, B1, and D2 if an insert **Text with Number** is set with a *%d* insert at position 0 from the *End*, regardless of the existence of an object C in the project hierarchy that was not added to the **Batch Rename**.

Related Topics

- [Opening the Batch Rename View](#)
- [Applying Batch Rename Changes](#)

Applying Batch Rename Changes

After [Specifying Batch Rename Settings](#), but before applying them, it is good practice to check the **Preview** panel for the impact of your specified rename settings.

To review the preview:

1. Check for a count of detected errors and warnings in the field just below the panel.
2. Sort the objects by the message type column (click the column header without a title, to the right of **After**).

Objects with error messages are listed first, with warning messages second, and without messages third.

3. Look through the objects with messages by inspecting the **Before** and **After** columns.

The **After** column displays the effect of the **Settings** panel entries.

Isolate what, if anything (warnings do not necessarily imply required changes), you need to adjust.

4. Change the **Settings** panel entries as desired, then click the **Preview** panel.

The panel updates to show the impact of your changes.

5. Continue adjusting the settings until there are no error messages and you are satisfied with the forecast results (in the **After** column).

To apply the rename settings to all the listed objects:

1. Click **Rename All** to change the object names or comments listed in the **Before** column of the **Preview** panel.



Caution

The **Batch Rename** applies to all the objects that were added in the **Preview** panel, even if some of those objects are filtered from view with a search filter.

All the listed object names or comments are changed to their **After** column values.

The message type and **Message** columns update to respectively display a green square and a **Successfully renamed** message.

Related Topics

- [Opening the Batch Rename View](#)
- [Specifying Batch Rename Settings](#)

audio**kinetic**

Part II. Setting Up Your Projects



3. Working with Projects	52
Overview	53
Managing Projects	55
Defining your Project Settings	60
Defining the Default User Settings for your Project	88
Troubleshooting Your Project	90
The Learning Annex - More on Obstruction and Occlusion	96
Project Management Tips and Best Practices	98
4. Managing Platforms	100
Overview	101
Platform Manager	101
PS Vita Hardware Versus Software Platforms	104
5. Managing Workgroups	107
Overview	108
Dividing Your Project into Work Units	111
Viewing the Status of Your Project Files	120
Using Wwise with Your Source Control System	122
Resolving Project Inconsistencies	123
Managing Project Files Using a Workgroup Plug-in	125
Workgroup Tips & Best Practices	138
6. Managing Media Files in Your Project	142
Overview	143
The Importing Process	143
Importing Media Files	151
Replacing Media Files	167
Managing File Import Issues	170
Re-Organizing Media Files within the Originals Folder	172
Clearing Your Cache	173
Editing Audio Files in an External Editor	174
Creating Audio and Motion Sources Using Plug-ins	176
Media File Management Tips and Best Practices	177
7. Building Your Actor-Mixer Hierarchy	178
Overview	179
About Properties in the Project Hierarchy	185
Building the Actor-Mixer Hierarchy	194
Enhancing Sound and Motion by Randomizing Property Values	200
Building Actor-Mixer Hierarchy Tips and Best Practices	201
8. Building the Structure of Output Busses	206
Overview	207
Defining the Properties of a Bus	212
Structuring a Bus Hierarchy - Example	217

Chapter 3. Working with Projects

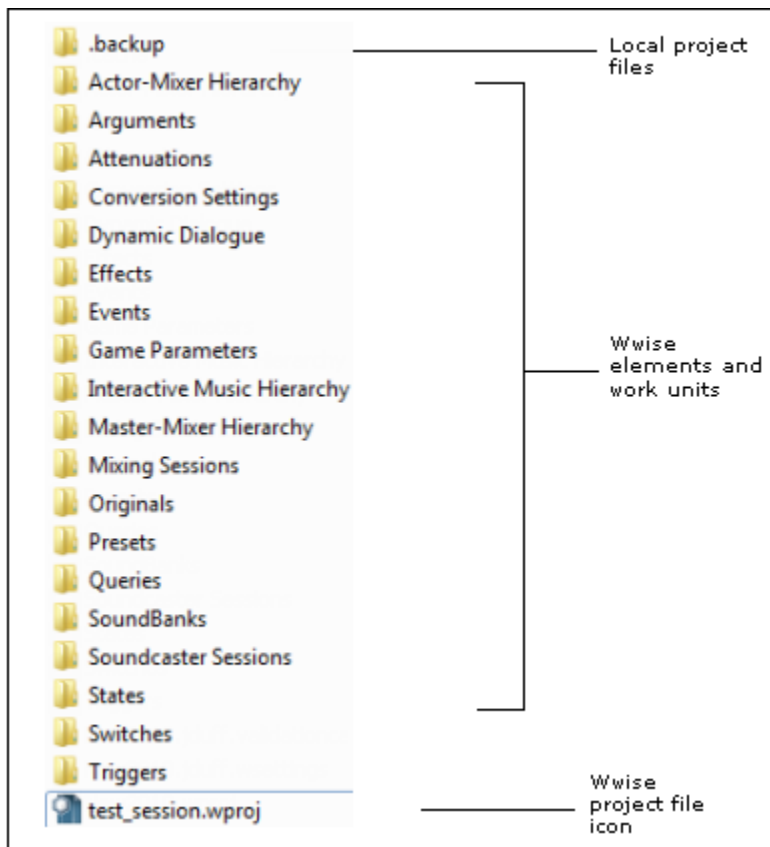
Overview	53
Managing Projects	55
Defining your Project Settings	60
Defining the Default User Settings for your Project	88
Troubleshooting Your Project	90
The Learning Annex - More on Obstruction and Occlusion	96
Project Management Tips and Best Practices	98

Overview

In Wwise, a project contains all your assets, and the properties and behaviors that you set for your assets for each platform and language that you are developing. That same project also contains Wwise elements such as events, presets, logs, and simulations as well as the SoundBanks that you will generate. A project contains all your work and, if you work with others, that of your colleagues as well.

You must use only one Wwise project per game. You can, however, divide up a project into separate work units. For information on using work units, refer to [Dividing Your Project into Work Units](#).

The starting point for developing your project in Wwise is the Project Launcher where you can create and open your project. When you create a project, a series of folders are created in the location that you choose on your workstation or the network.



The Wwise project folder structure consists of various folders containing XML files for the different project elements to make it easier to manage project versions and multiple users. A typical project folder contains the following:

- **Cache** - Converted versions of the SFX, Voice, and Plugin assets imported into your project. Do not include this folder in your source control system.
- **Actor-Mixer Hierarchy** - Default and user-created work units for the project's sound and motion structures.
- **Conversion Settings** - Default and user-created work units for the project's conversion settings ShareSets.
- **Dynamic Dialogue** - Default and user-created work units for the project's dialogue events.
- **Effects** - Default and user-created work units for the project effect ShareSets.
- **Events** - Default and user-created work units for the project events.
- **Game Parameters** - Default and user-created work units for game parameters.
- **Interactive Music Hierarchy** - Default and user-created work units for the project's music structures.
- **Master-Mixer Hierarchy** - Default and user-created work units for the project output routing.
- **Mixing Sessions** - Default and user-created work units for mixing sessions.
- **Originals** - Exact copies of the original versions of the SFX, Voice, and Motion assets imported into your project.
- **Presets** - Default and user-created work units for project presets.
- **Queries** - Default and user-created work units for Queries.
- **SoundBanks** - Default and user-created work units for SoundBanks. After you generate SoundBanks for your project, a new Generated SoundBanks folder is displayed.
- **Soundcaster Sessions** - Default and user-created work units for Soundcaster Sessions.
- **States** - Default and user-created work units for States.
- **Switches** - Default and user-created work units for Switches.
- **Triggers** - Default and user-created work units for Triggers.
- **.validationcache** - A list of all project files that have been validated against the current XML schema version. By keeping track of the validated files, Wwise no longer needs to validate the files each time it loads the project. This greatly reduces the time it takes Wwise to load a project. This file should not be managed by your source control system.
- **.wsettings** - Default conversion and miscellaneous object settings defined for the current project. These settings are saved per user. This file should not be managed by your source control system.
- **.wproj** - Wwise project file. Double-click the project icon to open your project.

After you have created your project, you can begin dividing up the work into work units and building asset structures based on the game design. At the same

time, you can also build the structure for your project routing in the Master-Mixer hierarchy, and create the project Game Syncs.

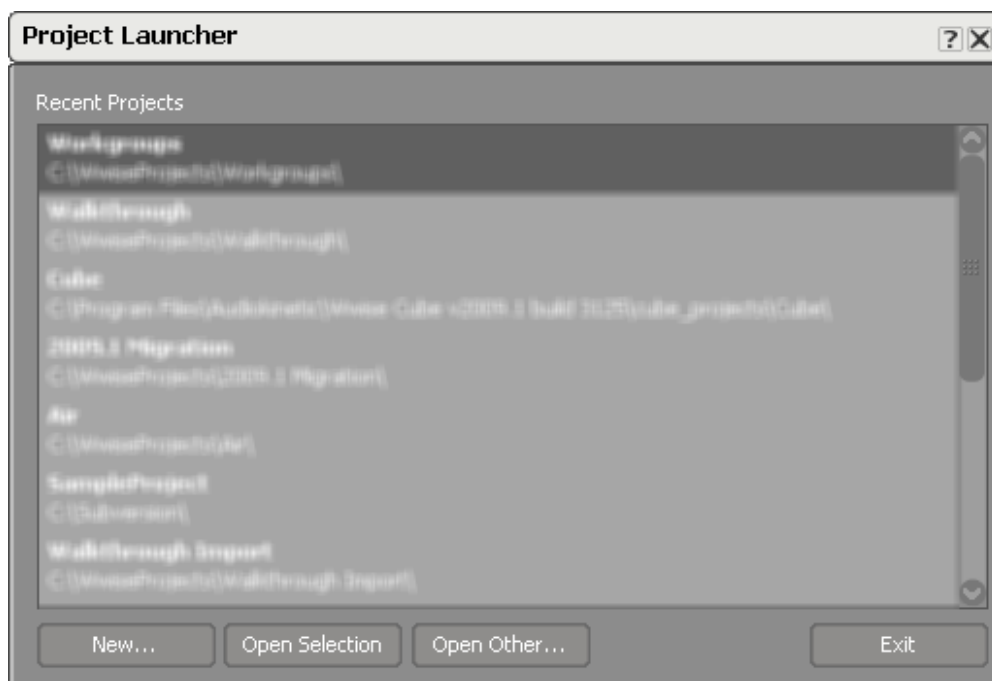
In addition to these folders in your project, you may also have other files such as integrity reports and project header files for SoundBanks.

The Wwise project and its folders have been designed so that it can be easily integrated into your external source control management tools. For more information about how to manage multiple project users, refer to [Chapter 5, *Managing Workgroups*](#).

Managing Projects

Since one Wwise project contains all the sound, music, motion assets, properties, and SoundBanks for your game audio and motion, it is very important to manage projects carefully. Like other Windows programs, you can manage your project folder and carry out standard tasks such as copying, moving, and deleting projects in Windows Explorer. To create and save a project, however, you need to work in Wwise.

When you open Wwise from the Start menu, the **Project Launcher** is displayed where you can either create a new project, or open an existing project.



After you have opened a project in Wwise, you can access a series of commands in the Project menu where you can open, close, create, save projects, and so on. Many of the commands have been mapped to keyboard shortcuts. For a complete list of shortcuts, refer to [Appendix B, *Shortcuts*](#).

Creating a New Project

You need to use the **Project Launcher** to open **New Project** and create your first Wwise project. Within a loaded Wwise project, **New Project** is accessed from the **Project > New...** menu option.

To create a new project:

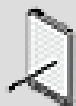
1. Do one of the following:

In the **Project Launcher**, click **New...**

From the Wwise menu bar, click **Project > New**.

Press **Ctrl + N**.

2. The **New Project** dialog opens.
3. Enter the name for the new project in **Name**.



Note

Each project name in Wwise must be unique, and the following characters may not be used: ‘:<>*?”\|.’.

As you type a valid project name, the project path is updated in **Project folder**, and **OK** activates.

4. In **Location**, do one of the following:

From the list of paths, select a path that was previously used for your projects.

Click **Browse [...]** to navigate to a location where you want to create your new project.

The path that you have selected for the project folder is displayed in **Project folder**.



Note

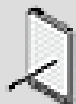
If you select a mapped network for your project, this may result in some audio playback performance issues that Audiokinetic will not be able to support.

5. In **Original files**, do one of the following:

Select **Use Default Originals Directory** from the list. This folder is located in the Wwise project folder.

Click **Browse [...]** to navigate to a location where you want to create your "Originals" folder.

6. Review the list of platforms. Click **Add...** for [Adding a Platform](#), **Remove** for [Removing a Platform](#), or **Rename** to change the name of any of the listed platforms you select.



Note

You can change your project's platforms at a later time in the **Platform Manager**, which can be opened from the **Projects** menu.

7. Select all the asset groups you would like to include in your project.
8. Click **OK**.

The new project is created in the Wwise projects folder in the location that you have specified.

Related Topics

- [Opening and Closing a Project](#)
- [Saving a Project](#)
- [Platform Manager](#)

Opening and Closing a Project

When you open Wwise, the projects you have worked on recently are included in the Recent Projects list. You can open one of these projects, or browse to open another project. Keep in mind that you can only have one project open at a time. If you already have one project open and you want to open another, you need to close the opened project first.

Before a project is opened, it is validated in Wwise. If there are XML syntax errors, the project will not load. If there are project inconsistencies, the Project Load Log dialog box will open with a list of inconsistencies along with possible fixes. If you accept these fixes suggested by Wwise, they are carried out but not saved until you save the project. For more information about how to deal with these project-related issues, refer to [Resolving Project Inconsistencies](#).

If you are working on a very large project, and you find it is taking a long time to load, you can unload specific work units, keeping on those that you are

working on. After these work units have been unloaded from the project, Wwise will no longer load them each time the project is opened. This can speed up the opening of a project significantly. You can easily re-load a work unit back into the project at any time. For more information on loading and unloading work units to and from your project, refer to [Loading/Unloading Work Units from Your Project](#).

To open a project from the Project Launcher:

1. Do one of the following:

In the Recent Projects list of the Project Launcher, select a project and click **Open Selection**.

Click **Open Other** to navigate to the project you want to open.

The Loading Project dialog box displays a progress bar, and then the selected project opens in the Designer Layout.



Tip

You can double-click a project in the Recent Projects list to open it.

To open a project from within Wwise:

1. Do one of the following:

From the menu bar, click **Project > Open**.

Press **Ctrl + O**.

The Open folder opens where you can browse to the project that you want to open.

2. Navigate to the project folder that contains the Wwise project file.
3. Select the .wproj file, and click **Open**.

The selected project opens.



Note

You can also open a project by navigating to the project folder and double-clicking the Wwise project icon. If you currently have more than one version of Wwise installed on your

system, the project icon always launches the latest version of Wwise.

To close a project:

1. Do one of the following:

From the menu bar, click **Project > Close**.

Press **Ctrl + F4**.

The project is closed and you are returned to the Project Launcher.



Note

If you have deleted objects in your project, the associated audio files will not be deleted when you close the project. To remove these files, you will need to clear your cache. For more information, refer to [Clearing Your Cache](#).

Related Topics

- [Creating a New Project](#)
- [Saving a Project](#)
- [Resolving Project Inconsistencies](#)
- [Loading/Unloading Work Units from Your Project](#)

Saving a Project

When you make any changes to your current project, an asterisk appears beside the name of the project in the title bar, as well as beside the project work units that have changed in the Project Explorer. It is a good idea to save your project on a regular basis.

In some cases, you may not be able to save parts of your project if one or more work units that you have modified are read-only. Before you begin working on a work unit, verify that you can save any changes that you might make. When using a source control system to manage your project files, keep in mind that you will need to check out your project files in order to save them. For more information on saving project files that are managed by a source control system, refer to [Saving Your Project When Using Perforce](#).



Note

Any fixes made to project inconsistency errors listed in the Project Load Log dialog box, will not be saved until you save the project. For more information about these errors, refer to [Resolving Project Inconsistencies](#).

To save a project:

1. Do one of the following:

From the Wwise menu bar, click **Project > Save**.

Press **Ctrl + S**.

The changes that you have made to your project are saved, and you are returned to the project.

Related Topics

- [Creating a New Project](#)
- [Opening and Closing a Project](#)

Defining your Project Settings

In Wwise, you can define certain settings for your project in the Project Settings dialog box.

The Project Settings dialog box includes the following tabs:

- **General** to define a source control plug-in, volume thresholds for each platform, the location for the Originals folder for your project assets, automatic sample rate detection settings, as well as event name creation settings.
- **Conversion** to set the default conversion settings ShareSet for the project as well as the automatic sample rate detection settings.
- **SoundBanks** to define the SoundBank settings for your project, which includes whether to generate content files, a header file, and the maximum attenuation information for events with your SoundBanks, whether SoundBank names can be used, and the location where your SoundBanks will be saved. Within this tab, you can also create custom steps that will be performed before and/or after the SoundBanks are generated.
- **Logs** to manage the warnings, errors, and messages that are displayed in the Conversion and SoundBank logs.
- **Obstruction/Occlusion** to define the volume, LPF and HPF curves for obstruction and occlusion for each platform in your project.

- [Motion Devices](#) to enable the motion devices for which you want to create motion effects for your game.
- [External Sources](#) to specify the input and output paths for the external audio sources that will be used with the External Source plug-in.
- [Network](#) to specify port numbers to be used during communication between the authoring application and your game.
- to define properties for the Sound and Audio Source objects of the Actor-Mixer Hierarchy.

Defining the General Settings for Your Project

In the General tab of the Project Settings dialog box, you can carry out the following tasks:

- [Specifying Volume Thresholds for a Project](#)
- [Configuring Source Control Plug-ins](#)
- [Defining Originals Folder Settings](#)
- [Defining Cache Folder Settings](#)
- [Defining Event Creation Settings](#)

Specifying Volume Thresholds for a Project

The volume threshold can be specified for each platform. The volume threshold refers to the point below which a voice will be managed by behaviors defined in the Advanced Settings tab of the Property Editor. You can specify a default volume threshold for each platform in your project in the Project Settings dialog box. This value will be used if the volume threshold is not specifically set using the Wwise API. For more information on managing low level sounds and motion objects and defining the object's behavior when it reaches the volume threshold, refer to [Managing Low-Volume Sounds and Motion Objects](#).

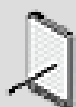
To define the volume thresholds for your project:

1. Open the Project Settings dialog box by doing one of the following:

From the Project menu, select **Project Settings**.

Press **Shift+K**.

2. In the Platforms group box, specify the point at which the sound will be managed by the behaviors defined in the Advanced Settings tab of the Property Editor.



Note

This default value will be used if no volume threshold setting is specified using the Wwise API.

3. Click **OK** to save your settings and to close the Project Settings dialog box.

Related Topics

- [Configuring Source Control Plug-ins](#)
- [Defining Originals Folder Settings](#)
- [Defining Cache Folder Settings](#)
- [Defining the Sample Rate Automatic Detection Settings](#)
- [Defining the SoundBank Settings for Your Project](#)
- [Managing Messages that Appear in the Logs](#)
- [Defining Obstruction and Occlusion Curves for Your Project](#)
- [Enabling the Motion Devices for Your Project](#)
- [Specifying the Input/Output Locations for External Sources](#)
- [Defining the Default User Settings for your Project](#)
- [Troubleshooting Your Project](#)
- [Defining Event Creation Settings](#)
- [Copying Settings from One Platform to Another](#)

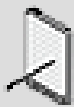
Configuring Source Control Plug-ins

If you are using a source control system to manage your project assets and files, you can select a Workgroup plug-in for your project. If the plug-in is supported in the current version of Wwise, you can also configure the plug-in for your workspace.

Wwise comes with the following two source control plug-ins:

- **Perforce®** - Refer to [Perforce](#) for more information about the compatible versions.
- **Subversion** - Refer to [Subversion](#) for more information about the compatible versions.

For more information on what versions of Perforce and Subversion are supported by Wwise, refer to [Supported Perforce/Subversion Versions](#) or the [Wwise SDK documentation](#).



Note

Each source control plug-in will have different requirements for the configuration. Verify the configuration settings with your system administrator.

To define the source control plug-in for your project:

1. Open the Project Settings dialog box by doing one of the following:

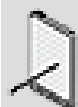
From the Project menu, select **Project Settings**.

Press **Shift+K**.

2. From the Plug-in list of the Workgroup group box, select the Source Control plug-in for your project.
3. To configure the plug-in, click **Config**.

The Source Control Plug-in Configuration dialog box opens.

4. Enter the required information in the fields of the Source Control Plug-in Configuration dialog box.



Note

Verify the configuration settings with your system administrator.

5. Click **OK** to save your configuration settings and to close the Plug-in Configuration dialog box.
6. Click **OK** to save your settings and close the Project Settings dialog box.

Related Topics

- [Specifying Volume Thresholds for a Project](#)
- [Defining Originals Folder Settings](#)
- [Defining Cache Folder Settings](#)
- [Defining the Sample Rate Automatic Detection Settings](#)
- [Defining the SoundBank Settings for Your Project](#)
- [Managing Messages that Appear in the Logs](#)
- [Defining Obstruction and Occlusion Curves for Your Project](#)
- [Enabling the Motion Devices for Your Project](#)
- [Specifying the Input/Output Locations for External Sources](#)
- [Defining the Default User Settings for your Project](#)
- [Troubleshooting Your Project](#)
- [Defining Event Creation Settings](#)

Defining Originals Folder Settings

The project's Originals folder contains copies of the original audio files that you import into your project. Its location is defined when you create the project. In the General tab, you can specify the location for the Original files. You can select a location for the entire project, or you can choose to save these files in an alternate location for your own use. This option can be very useful in the following situations:

- You are working remotely and don't have access to the Originals folder.

- You do not have permission to alter the contents of the Originals folder.
- You need to create a temporary location for your Originals folder without changing the location of the project Originals folder.



Note

If the project-defined Originals folder is not accessible, the override setting will be automatically turned on, and the Originals folder will be moved to your project folder so that you can work in the project.

For more information about the Originals folder, refer to [The Media File Structure](#).

To define the location of the Originals folder for your project:

1. Open the Project Settings dialog box by doing one of the following:

From the Project menu, select **Project Settings**.

Press **Shift+K**.

2. To define the Originals file folder location for your project, do one of the following:

In the Original Audio Files group box, type or paste the path of the Original folder. This path can be absolute or relative to the project's folder. The default path is "Originals".

Click the **Browse** button (...) to navigate to a location where you want to store your Originals audio file folder.

3. To define the Originals file folder location for your personal use, do one of the following:

In the Original Audio Files group box, select the **Override location for current user** option.

Click the **Browse** button (...) to navigate to a location where you want to store your Originals audio file folder.

4. Click **OK** to save your settings and close the Project Settings dialog box.

Related Topics

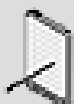
- [Specifying Volume Thresholds for a Project](#)
- [Configuring Source Control Plug-ins](#)
- [Defining Cache Folder Settings](#)
- [Defining the Sample Rate Automatic Detection Settings](#)

- [Defining the SoundBank Settings for Your Project](#)
- [Managing Messages that Appear in the Logs](#)
- [Defining Obstruction and Occlusion Curves for Your Project](#)
- [Enabling the Motion Devices for Your Project](#)
- [Specifying the Input/Output Locations for External Sources](#)
- [Defining the Default User Settings for your Project](#)
- [Troubleshooting Your Project](#)
- [Defining Event Creation Settings](#)

Defining Cache Folder Settings

The project's cache folder contains intermediate data generated by Wwise during audio file conversion and SoundBank generation. Its location is initially set to '.cache/' inside the project's directory when a new project is created. On the General tab of the Project Settings dialog box, you can modify the location for the cache files. You can select a location for the entire project, or you can choose to save these files in an alternate location for your own use. This option can be useful in the following situations:

- You are working remotely and don't have access to the cache folder.
- You do not have permission to alter the contents of the cache folder.
- You need to create a temporary location for your cache folder without changing the location of the project cache folder.



Note

If the project-defined cache folder is not accessible, the override setting will automatically be turned on, and the cache folder will be moved to your project folder so that you can work in the project.

If you chose to override the location of the cache folder, the next time you open your Wwise project it will be in the new location. Other users, however, will still access the folder in its original location. Turn off this option to go back to accessing the cache folder at the project location. For more information about the cache folder, refer to [The Media File Structure](#).



Caution

Multiple users should NOT access the same cache folder simultaneously.

To define the location of the cache folder for your project:

1. Open the Project Settings dialog box by doing one of the following:

From the Project menu, select **Project Settings**.

Press **Shift+K**.

2. To define the cache file folder location for your project, do one of the following:

In the Cached Audio Files group box, type or paste the path of the cache folder. This path can be absolute or relative to the project's folder. The default path is ".cache".

Click the **Browse** button (...) to navigate to a location where you want to store your cache file folder.

3. To define the cache file folder location for your personal use, do one of the following:

In the Cached Audio Files group box, select the **Override location for current user** option.

Click the **Browse** button (...) to navigate to a location where you want to store your cache file folder.

4. Click **OK** to save your settings and close the Project Settings dialog box.

Related Topics

- [Specifying Volume Thresholds for a Project](#)
- [Configuring Source Control Plug-ins](#)
- [Defining Originals Folder Settings](#)
- [Defining the Sample Rate Automatic Detection Settings](#)
- [Defining the SoundBank Settings for Your Project](#)
- [Managing Messages that Appear in the Logs](#)
- [Defining Obstruction and Occlusion Curves for Your Project](#)
- [Enabling the Motion Devices for Your Project](#)
- [Specifying the Input/Output Locations for External Sources](#)
- [Defining the Default User Settings for your Project](#)
- [Troubleshooting Your Project](#)
- [Defining Event Creation Settings](#)

Defining Event Creation Settings

Events in Wwise can be created in a multitude of ways. If an event is created without a target object, Wwise will give the new event a generic name. If however the event is created for a selected object, you can specify how Wwise constructs the new event's name.

To define the Event Creation Settings for your project:

1. Open the Project Settings dialog box by doing one of the following:

From the Project menu, select **Project Settings**.

Press **Shift+K**.

2. Select whether to specify the Event Creation Settings for the project or the current user:

To set the Event Creation Settings for the project, select **Define settings for project**.

To set the Event Creation Settings for the current user, select **Override settings for current user**.

3. Select the desired Event Creation Settings:

Enable **Add action name** to include the action name in the event name.

Enable **Modify case** to set the case of the event name.

See below for more details.

4. Click **OK** to save your settings and close the Project Settings dialog box.

By default, the object's name is used as the new event name. The following options can modify the new event name:

- **Add action name**: if enabled, the event's action name is added to the event name.
 1. *set as prefix* - the action name is used as a prefix to the object name:
<action_name>_<object_name>
 2. *set as suffix* - the action name is used as a suffix to the object name:
<object_name>_<action_name>
- **Modify case**: if enabled, the case of the event's name is changed.
 1. *all lowercase* - the event's name is all lowercase.
 2. *all uppercase* - the event's name is all uppercase.



Note

The Event Creation Settings may be configured for the entire project, or for the current user.

Related Topics

- [Specifying Volume Thresholds for a Project](#)
- [Configuring Source Control Plug-ins](#)

- [Defining Originals Folder Settings](#)
- [Defining the Sample Rate Automatic Detection Settings](#)
- [Defining the SoundBank Settings for Your Project](#)
- [Managing Messages that Appear in the Logs](#)
- [Defining Obstruction and Occlusion Curves for Your Project](#)
- [Enabling the Motion Devices for Your Project](#)
- [Specifying the Input/Output Locations for External Sources](#)
- [Defining the Default User Settings for your Project](#)
- [Troubleshooting Your Project](#)
- [Defining Cache Folder Settings](#)

Defining the Conversion Settings for Your Project

On the Source Settings tab of the Project Settings dialog box, you can carry out the following tasks:

- [Specifying the Default Conversion Settings](#)
- [Defining the Sample Rate Automatic Detection Settings](#)

Specifying the Default Conversion Settings

After you have created the conversion settings ShareSets for your project, you can specify which one you want to use as the default.

The default conversion settings ShareSet is used in the following situations:

- When a new object is created. The default ShareSet will only be used if the new object is a top-level parent object. If the new object is a child of another object, it will inherit the conversion settings assigned to the parent.
- During SoundBank generation. If an object has not been assigned a conversion settings ShareSet, the default ShareSet will be used to convert the object before the SoundBanks are generated.

To specify the default conversion settings ShareSet:

1. Open the Project Settings dialog box by doing one of the following:

From the Project menu, select **Project Settings**.

Press **Shift+K**.

2. Switch to the **Source Settings** tab.
3. In the Default Conversion Settings group box, click the **Browse** button (...).

The Project Explorer - Browser opens.

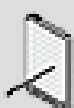
4. Select the ShareSet that you want to use for your project's default conversion settings.
5. Click **OK**.

Related Topics

- [Specifying Volume Thresholds for a Project](#)
- [Configuring Source Control Plug-ins](#)
- [Defining the Sample Rate Automatic Detection Settings](#)
- [Defining the SoundBank Settings for Your Project](#)
- [Managing Messages that Appear in the Logs](#)
- [Defining Obstruction and Occlusion Curves for Your Project](#)
- [Enabling the Motion Devices for Your Project](#)
- [Specifying the Input/Output Locations for External Sources](#)
- [Defining the Default User Settings for your Project](#)
- [Troubleshooting Your Project](#)

Defining the Sample Rate Automatic Detection Settings

Deciding on the best sample rate for your media files can be difficult and time-consuming. To help speed up the process, you can have Wwise perform an analysis of each file using the Fast Fourier Transform (FFT) algorithm. At a very basic level, Wwise uses the FFT to generate a spectral analysis of the media file by analyzing the sound wave one portion at a time using a Hanning window. A cutoff volume or threshold identifies a frequency that is used to determine the best sample rate at which to convert your files. As part of your project settings, you can define the size of the Hanning window used by the FFT algorithm as well as the threshold levels for three different quality settings: High, Medium, and Low. These threshold settings are used when you select Auto High, Auto Medium, or Auto Low as your sample rate conversion method.



Note

The volume threshold values are used in the context of a normalized spectrum.

To define the automatic sample rate detection settings for your project:

1. Open the Project Settings dialog box by doing one of the following:

From the Project menu, select **Project Settings**.

Press **Shift+K**.

2. From the FFT window size list in the Sample Rate Automatic Detection group box, select the size of the Hanning window that will be used by the FFT algorithm to analyze the sound wave.

3. In the Volume Thresholds group box, specify the cutoff volume level used by the FFT algorithm for each of the following quality options:

Low quality - A cutoff volume level that identifies the frequency that is used to determine the best sample rate at which to convert your files. The low quality threshold is used when the Auto Low option is selected as your conversion sample rate.

Medium quality - A cutoff volume level that identifies the frequency that is used to determine the best sample rate at which to convert your files. The medium quality threshold is used when the Auto Medium option is selected as your conversion sample rate.

High quality - A cutoff volume level that identifies the frequency that is used to determine the best sample rate at which to convert your files. The high quality threshold is used when the Auto High option is selected as your conversion sample rate.



Note

A higher threshold level will result in a lower quality sample rate being used in the conversion process and a smaller file size.

4. Click **OK** to save your settings and close the Project Settings dialog box.

Related Topics

- [Specifying Volume Thresholds for a Project](#)
- [Configuring Source Control Plug-ins](#)
- [Defining Originals Folder Settings](#)
- [Defining Cache Folder Settings](#)
- [Defining the SoundBank Settings for Your Project](#)
- [Managing Messages that Appear in the Logs](#)
- [Defining Obstruction and Occlusion Curves for Your Project](#)
- [Enabling the Motion Devices for Your Project](#)
- [Specifying the Input/Output Locations for External Sources](#)
- [Defining the Default User Settings for your Project](#)
- [Troubleshooting Your Project](#)

Defining the SoundBank Settings for Your Project

Before generating the SoundBanks for each of your platforms and languages, you need to define the settings for your SoundBanks. The following SoundBank settings can be defined at the project level:

- [Defining SoundBank Project Settings](#) - determine what information is to be included with the generated SoundBanks.
- [Specifying a Location for Your Saved SoundBanks](#) - determine the location on your hard drive or network where the SoundBanks will be saved.
- [Defining Steps to be Performed Pre/Post SoundBank Generation](#) - determine which tasks will be performed immediately before the SoundBanks are generated.
- [Defining Steps to be Performed Pre/Post SoundBank Generation](#) - determine which tasks will be performed immediately after the SoundBanks are generated.

Although these settings are defined at the project level, you can create custom user settings by overriding these project settings. For more information on overriding the SoundBank project settings, refer to [Defining Custom Attributes for Your SoundBanks](#).

Defining SoundBank Project Settings

Before generating your SoundBanks, you need to determine what information will be part of the generation process, how it will be included, and in what format will it be generated. The settings you choose will depend on how the data and media within the SoundBanks are accessed by your game.

To define SoundBank project settings:

1. Open the Project Settings dialog box by doing one of the following:

From the Project menu, select **Project Settings**.

Press **Shift+K**.

2. Switch to the **SoundBanks** tab.
3. In the SoundBank Settings group box, select any of the following options to define custom settings for your SoundBanks:

Allow SoundBanks to exceed maximum size to generate SoundBanks even if they exceed the maximum size specified.

Generate SoundBank content files to create files that list the contents of each SoundBank. The content files include information on events, busses, states, and switches, as well as a complete list of streamed and in memory audio files.

Generate header file to create a header file that maps event, state, switch, and game parameter names to IDs.

Max attenuation to include maximum attenuation information in the SoundBanksInfo.xml file for each event.

Estimated duration to include the estimated maximum and minimum duration for each event, as well as whether a sound loops infinitely or is a one-shot sound, in the SoundBanksInfo.xml file for each event.

Use **SoundBank Names** to use SoundBank names (checked) or IDs (unchecked) to name generated .bnk SoundBank files and to reference one bank within another bank.

4. If you chose to generate a header file, you must decide where it will be saved. To do so, do the following:

Type a path directly in the text box.

Click the **Browse** button (...) and use the browser to navigate to the location of your choice.



Note

You can use a full path or a relative path to specify the location where the header file will be saved. When using a relative path, use the project folder as the origin of the path.

5. If you chose to generate SoundBank content files, you can select the desired text file format with the **SoundBank content file format** option.



Tip

If you have file paths, object names or object notes that contain non-ANSI characters, you should use the Unicode format.

6. Click **OK** to apply the settings.

Related Topics

- [Specifying a Location for Your Saved SoundBanks](#)
- [Defining Steps to be Performed Pre/Post SoundBank Generation](#)
- [Defining the General Settings for Your Project](#)
- [Managing Messages that Appear in the Logs](#)
- [Defining Obstruction and Occlusion Curves for Your Project](#)
- [Enabling the Motion Devices for Your Project](#)
- [Specifying the Input/Output Locations for External Sources](#)
- [Defining the Default User Settings for your Project](#)

Specifying a Location for Your Saved SoundBanks

When you generate the SoundBanks for your project, they are saved by default in the following folder:

ProjectName\GeneratedSoundBanks\Platform\

If this location is not convenient for you, you can change it to any directory on your workstation or network.

When specifying the location for your saved SoundBanks, you can use a full path or a relative path. When using a relative path, use the project folder as the origin of the path. For example, the following full path and relative path specify the same location:

- C:\Wwise Projects\My Project\GeneratedSoundBanks\Windows
- GeneratedSoundBanks\Windows\

To specify a new location for your saved SoundBanks:

1. Open the Project Settings dialog box by doing one of the following:

From the Project menu, select **Project Settings**.

Press **Shift+K**.

2. Switch to the **SoundBanks** tab.
3. In the SoundBank Paths group box, specify a path by doing one of the following:

Type a path directly in the text box.

Click the **Browse** button (...) and use the browser to navigate to the location of your choice.

4. Click **OK** to apply any changes you made.

Related Topics

- [Defining SoundBank Project Settings](#)
- [Defining Steps to be Performed Pre/Post SoundBank Generation](#)
- [Defining the General Settings for Your Project](#)
- [Managing Messages that Appear in the Logs](#)
- [Defining Obstruction and Occlusion Curves for Your Project](#)
- [Enabling the Motion Devices for Your Project](#)
- [Specifying the Input/Output Locations for External Sources](#)
- [Defining the Default User Settings for your Project](#)



Defining Steps to be Performed Pre/Post SoundBank Generation


Depending on your workflow, you may have a certain step or task that needs to be performed immediately before or immediately following the generation of your SoundBanks. For example, you may want to check out specific SoundBank files from your source control system before generating them or you may want the streamed files to be copied to the SoundBanks directory immediately following generation.

In Wwise, these types of tasks are defined by creating command lines. A special command line editor exists within Wwise making it easy for you to build as many command lines as you need. To simplify the process even further, the editor contains a list of all the Wwise-specific and other Windows environmental variables that can be used in a command line.

The specific Wwise variables available for writing custom command lines are as follows:

Command Line Variable	Description
\$(AllowExceedMaximum)	Specifies whether SoundBanks can be generated even if they exceed the maximum size specified. This variable is set to true when the Allow SoundBanks to exceed maximum option is selected.
\$(ContentFileFormat)	Specifies the file type for generated SoundBank content files. Possible values are: <ul style="list-style-type: none"> • ANSI • Unicode
\$(GenerateContentFile)	Specifies whether files that list the contents of each SoundBank are created. The content files include information on events, busses, states, and switches, as well as a complete list of streamed and in memory audio files. This variable is set to true when the Generate SoundBank content files option is selected.
\$(GenerateHeaderFile)	Specifies whether a header file is generated that maps event, state, switch, and game parameter names to IDs. This variable is set to true when the Generate header file option is selected.
\$(GenerateMaxAttenuationInfo)	Specifies whether the maximum attenuation information is generated for events. This variable is set to true when the Event Info: Max attenuation option is selected.
\$(GenerateEstimatedDuration)	Specifies whether the estimated maximum and minimum duration and duration type information is generated for events. This variable is set to true when the Event Info: Estimated Duration option is selected.
\$(HeaderFileFullPath)	The full path of the header file, which is: \$(HeaderFilePath)\Wwise_IDs.h
\$(HeaderFilePath)	The path or location where the header file will be saved.

Command Line Variable	Description
	This path is taken from the Header file path text box.
\$(InfoFilePath)	The full file name of the current platform's Info file.
\$(IsRunningFromCmdLine)	Specifies whether Wwise was launched from the command line with the “-generatesoundbanks” flag.
\$(LanguageList)	The list of languages passed to the command line OR the selected languages in the SoundBank Manager. <div style="border: 1px solid gray; background-color: #f0f0f0; padding: 5px; margin-top: 10px;">  <p>Note The list is a space-separated list.</p> </div>
\$(Platform)	The name of the current platform.
\$(SoundBankList)	The list of SoundBanks passed to the command line OR the selected SoundBanks in the SoundBank Manager. <div style="border: 1px solid gray; background-color: #f0f0f0; padding: 5px; margin-top: 10px;">  <p>Note The list is a space-separated list. Use double-quotes to enclose the list in one single argument.</p> </div>
\$(SoundBankPath)	The path or location where the current platform's SoundBanks are saved.
\$(UseSoundBankNames)	Specifies whether SoundBank names (true) or IDs (false) are used to name generated soundbank .bnk files, as well as within banks to refer to media in other banks. This variable is set to true when the Use SoundBank names option is selected.
\$(WwiseExeDriveLetter)	The drive letter on your workstation where the Wwise executable (Wwise.exe) is located.
\$(WwiseExePath)	The path or location of the Wwise executable (Wwise.exe).
\$(WwiseExeProcessID)	The numerical Process Identifier of the Wwise executable (Wwise.exe).
\$(WwiseProjectDriveLetter)	The drive letter on your workstation where the Wwise project is located.
\$(WwiseProjectName)	The name of the current project.
\$(WwiseProjectPath)	The path or location of the Wwise project.



Note
Environment variables are automatically mapped, for example, \$(WWISESDK).

To be as flexible as possible, Wwise allows to define different command lines for the following types of steps:

- **Global opening step** - A command line that applies to all platforms and is performed before any other step.
- **Platform-specific pre-generation step** - A command line that applies to a specific platform and is performed before the SoundBanks are generated.
- **Platform-specific post-generation step** - A command line that applies to a specific platform and is performed after the SoundBanks are generated.
- **Global closing step** - A command line that applies to all platforms and is performed after all other steps.

By default, every project includes a platform-specific post-generation step command line that copies the streamed files to the SoundBank directory. You can, however, automate any type of task by executing a different command line. Wwise also ships with another Factory command line that uses the File Packager to generate a package containing all data and media within your SoundBanks. For more information about the File Packager, refer to [Chapter 35, *Managing File Packages*](#). For more information about loading factory command lines, refer to [Loading Factory/Custom Command Lines](#).

You can also save the command lines you create to a file (.wcmdline) so that you can use them later on, within the same project, across projects, or if you want to share them with other users. For more information on saving commands, refer to [Saving Custom Command Lines to a File](#).

To define tasks to be performed pre SoundBank generation:

1. Open the Project Settings dialog box by doing one of the following:

From the Project menu, select **Project Settings**.

Press **Shift+K**.

2. Switch to the **SoundBanks** tab.
3. To create a global pre-generation step, click the Edit button (...) for the Global opening step.

The Pre-Generation Step Editor opens.

4. In the Description text box, type a name that clearly describes the step or task that will be performed.
5. Click in the Commands text box to begin creating your command line.



Note

The Commands text box works like most other text editors, which means you can add new lines of text by pressing Enter, delete text by selecting it and pressing Delete, and so on.

6. If you want to insert built-in macros and environment variables in your command, do the following:

In the Macros group box, select one of the following options:

Built-in Macros - To display a list of Wwise-specific variables that can be used within the Wwise command lines.

Environment Variables - To display a list of Windows-specific environment variables that can be used within the Wwise command lines.

To add a variable to the command line, do one of the following:

Double-click a variable in list.

Select a variable from the list and then click **Insert**.

Continue to add variables to your command line, as required.

7. If you need to perform a second global pre-generation step, simply go to the end of the first line, press **Enter**, and then start creating a new command line.
8. Click **OK** to save the command line and to close the Pre-Generation Step Editor.



Note

If you want to save the command line to file, click the Save As button in the Editor. For more information on saving custom command lines, refer to [Saving Custom Command Lines to a File](#).

9. To create a platform-specific pre-generation step, repeat steps 3-8 for each platform.



Note

You can load factory and previously saved custom command lines into the Editor by clicking the Load button. For more information on loading factory/custom commands, refer to [Loading Factory/Custom Command Lines](#).

To define tasks to be performed post SoundBank generation:

1. Open the Project Settings dialog box by doing one of the following:

From the Project menu, select **Project Settings**.

Press **Shift+K**.

2. Switch to the **SoundBanks** tab.
3. In the Post-Generation Step group box, you will notice that the “Copy Streamed Files” command line is added by default. To modify this command line or to add an additional one, click one of the **Edit** buttons (...).

The Post-Generation Step Editor opens.

4. In the Description text box, type a name that clearly describes the step(s) or task(s) that will be performed.
5. In the Commands text box, click at the end of the current command line and press **Enter**. You can now begin to create a new command line.



Note

The Commands text box works like most other text editors, which means you can add new lines of text by pressing **Enter**, delete text by selecting it and pressing **Delete**, and so on.

6. If you want to insert built-in macros and environment variables in your command, do the following:

In the Macros group box, select one of the following options:

Built-in Macros - To display a list of Wwise-specific variables that can be used within the Wwise command lines.

Environment Variables - To display a list of Windows-specific environment variables that can be used within the Wwise command lines.

To add a variable to the command line, do one of the following:

Double-click a variable in list.

Select a variable from the list and then click **Insert**.

Continue to add variables to your command line, as required.

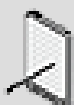
7. If you need to perform an additional pre-generation step, simply go to the end of the first line, press **Enter**, and then start creating a new command line.
8. Click **OK** to save the command line and to close the Post-Generation Step Editor.



Note

If you want to save the command line to file, click the Save As button in the Editor. For more information on saving custom command lines, refer to [Saving Custom Command Lines to a File](#).

9. Repeat steps 3-8 for the global closing step and/or for each additional platform.



Note

You can load factory and previously saved custom command lines into the Editor by clicking the Load button. For more information on loading factory/custom commands, refer to [Loading Factory/Custom Command Lines](#).

Related Topics

- [Loading Factory/Custom Command Lines](#)
- [Saving Custom Command Lines to a File](#)
- [Defining SoundBank Project Settings](#)
- [Specifying a Location for Your Saved SoundBanks](#)
- [Defining the General Settings for Your Project](#)
- [Managing Messages that Appear in the Logs](#)
- [Defining Obstruction and Occlusion Curves for Your Project](#)
- [Enabling the Motion Devices for Your Project](#)
- [Specifying the Input/Output Locations for External Sources](#)
- [Defining the Default User Settings for your Project](#)

Loading Factory/Custom Command Lines

Wwise ships with a few command lines that have already been created, including one that copies the streamed files to the SoundBank directory, and one that groups the streamed files, loose media, and SoundBanks into a package. These are called Factory command lines. You can load these factory command lines or custom command lines that you previously saved to file.

To load a factory/custom command line:

1. From the Pre/Post Generation Step Editor, click **Load**.
2. From the shortcut menu, click one of the following:

From Factory Folder - To open the browser to the directory where the Wwise Factory command lines are located.

From Last Location - To open the browser to the directory from which you last loaded a command line.

The Windows Open File dialog box opens.

3. Select the command line you want to load and click **Open**.

The command line is loaded into the Editor.

Related Topics

- [Saving Custom Command Lines to a File](#)
- [Defining Steps to be Performed Pre/Post SoundBank Generation](#)

Saving Custom Command Lines to a File

You can save the custom command lines you create to file so that you can use them later within the same project, across projects, or if you want to share them with other users.

To save a command line to file:

1. In the Pre-Post Generation Step Editor, write your command line(s).
2. When you are finished, click **Save As**.

The Save As dialog box opens.

3. Navigate to the folder where you want to save your command line, name it, and then press **Save**.

The command line is saved as a .wcmdline file and can now be re-used at any time.

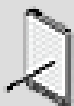
Related Topics

- [Loading Factory/Custom Command Lines](#)
- [Defining Steps to be Performed Pre/Post SoundBank Generation](#)

Managing Messages that Appear in the Logs

There may be situations where you no longer want or need specific warnings and/or messages to be displayed in the SoundBank or Conversion logs during the generation process. To prevent these types of warnings and messages from appearing in the SoundBank Log, you can add them to the Log Ignore list.

When you want to include these messages back in the log, you can simply remove the message types from the Log Ignore List. This list of messages is managed at the project level in the Project Settings dialog box.



Note

You can also add messages to the Log Ignore List by right-clicking a message directly in the log and selecting Add to Log Ignore List. For more information, refer to [Adding Messages to the Log Ignore List](#).

To add messages to the Log Ignore List:

1. From the menu bar, click **Project > Project Settings**.

The Project Settings dialog box opens.

2. Switch to the **Logs** tab.
3. Select the messages that you want to add to the Log Ignore List.
4. Click **OK** to close the Project Settings dialog box.

The next time a log is generated, the selected message types will be ignored by Wwise and will not be displayed in the log.



Note

To remove a message type from the Log Ignore List, simply remove the Ignore check mark for the corresponding message type.

To change the severity of a message:

1. From the menu bar, click **Project > Project Settings**.

The Project Settings dialog box opens.

2. Switch to the **Logs** tab.
3. Change the severity by using the drop down control on the *Severity* column.
4. Click **OK** to close the Project Settings dialog box.

The next time a log is generated, the selected severity will be applied.

To limit the number of messages that will appear in the logs:

1. From the menu bar, click **Project > Project Settings**.

The Project Settings dialog box opens.

2. Switch to the **Logs** tab.
3. Select the Limit number of messages displayed to: option and type the maximum number of messages you want to be displayed in the log. Note that this number is used to limit all message types in the log.
4. Click **OK** to close the Project Settings dialog box.

Related Topics

- [Defining the General Settings for Your Project](#)
- [Defining the SoundBank Settings for Your Project](#)
- [Defining Obstruction and Occlusion Curves for Your Project](#)
- [Enabling the Motion Devices for Your Project](#)
- [Specifying the Input/Output Locations for External Sources](#)

Defining Obstruction and Occlusion Curves for Your Project

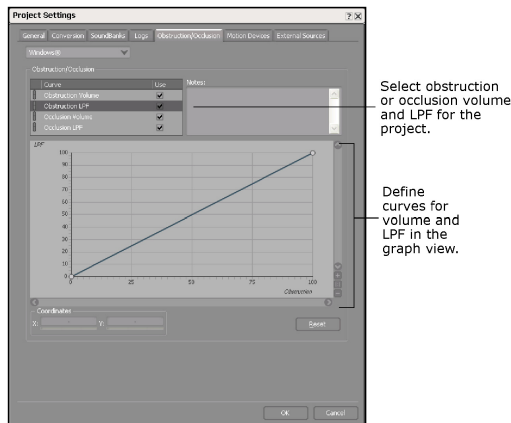
In the Obstruction/Occlusion tab of the Wwise Project Settings dialog box, you can define the obstruction and occlusion settings for the sound objects for each platform in your project. Obstruction occurs when an object in the game geometry, such as a wall or pillar, partially blocks the space between a sound source and a listener. Occlusion occurs when an object in the game geometry completely blocks the space between a sound source and its listener.

Game developers programmatically define the geometry of the game where the conditions for obstruction and occlusion may occur. Wwise does not compute obstruction and occlusion levels by itself. The physics calculations must be done by the game and the results must be passed to the following function:

```
SetObjectObstructionAndOcclusion()
```

For more information about how developers program obstruction and occlusion, refer to the [Obstruction and Occlusion in Environments](#) section in the Wwise SDK documentation.

Wwise allows you to define platform-specific volume and LPF curves that will be applied to sound objects as they are influenced by obstruction and occlusion during gameplay. The values you define are relative, and are added to any volume and LPF values already applied to the game object.



To create more detailed and complex obstruction and occlusion curves, you can define the shape of each curve segment. A curve segment is any part of the curve between two control points. You can choose from a variety of curve shapes, including linear, constant, logarithmic, exponential, and s-curve.

For more background information on obstruction and occlusion, refer to [The Learning Annex - More on Obstruction and Occlusion](#).



Note

LPF and HPF obstruction curves are not available for the Wii™ platform.

To define the obstruction and occlusion settings for your project:

1. Open the Project Settings dialog box by doing one of the following:
 - From the menu bar, click **Project > Project Settings**.
 - Press **Shift+K**.
2. Switch to the **Obstruction/Occlusion** tab.
3. From the Platform list, select the platform for which you are defining curves for the obstruction and occlusions settings.
4. To specify different settings for obstruction and/or occlusion volume, LPF and HPF on the selected platform, right-click the link indicator and in the shortcut menu, select **Unlink**.



The indicator will turn orange and the settings that you define for the unlinked properties will be used only on the selected platform.

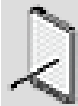
5. To define volume, LPF and HPF curves for the obstruction and occlusion settings for your game objects, do the following:

Select the Use check box to highlight the corresponding curve.

To create points on the curve, double-click a point on the curve.

To delete a point on the curve, select the point and press **Delete**.

To delete all the points you have defined so far and start over, click **Reset**.



Note

For information on zooming and panning the graph view, adding, moving, and deleting control points, changing the shape of the curve between points, using linear and dB scaling, and other general information about the Graph view, refer to [Chapter 42, *Getting to Know the Graph View*](#).

6. To annotate the curves, click in the Notes box and type your note.
7. When you are finished, click **OK**.

The Project Settings dialog box closes and your project obstruction/occlusion property curves are saved.

Related Topics

- [Zooming and Panning the Graph View](#)
- [Defining the Scaling Method of the Graph View](#)
- [Specifying the Shape of the Curve Between Control Points](#)
- [Creating a New Project](#)
- [Troubleshooting Your Project](#)
- [Defining the Default User Settings for your Project](#)
- [Defining the General Settings for Your Project](#)
- [Defining the SoundBank Settings for Your Project](#)
- [Managing Messages that Appear in the Logs](#)
- [Enabling the Motion Devices for Your Project](#)
- [Specifying the Input/Output Locations for External Sources](#)

Enabling the Motion Devices for Your Project

Before creating motion effects in your project, you need to enable the types of motion devices for which you want to create motion effects. After a motion device is enabled, you can do the following in your project:

- Create sources for Motion FX objects using media files or signal generators.

- Include motion data for the selected device in your SoundBanks.



Note

If you plan to develop, integrate, and distribute Wwise Motion with your game, you need to purchase a separate licence. For more information, contact the Audiokinetic sales team at: sales@audiokinetic.com.

To enable motion devices for your project:

1. Open the Project Settings dialog box by doing one of the following:

From the menu bar, click **Project > Project Settings**.

Press **Shift+K**.

2. Switch to the **Motion Devices** tab.
3. From the Devices list, select one or more of the following devices:

Controller - To create motion effects for each platform's game controller.

4. Click **OK**.

The Project Settings dialog box closes. You are now ready to begin creating the motion effects for your project.

Related Topics

- [Defining the General Settings for Your Project](#)
- [Defining the SoundBank Settings for Your Project](#)
- [Managing Messages that Appear in the Logs](#)
- [Defining Obstruction and Occlusion Curves for Your Project](#)
- [Specifying the Input/Output Locations for External Sources](#)
- [Defining the Default User Settings for your Project](#)

Specifying the Input/Output Locations for External Sources

If you plan to use the External Source plug-in you must specify the location of the External Sources List file. This file is a very simple XML file that specifies the following:

- The location of the external audio files that can be associated with the 'template' that you created in Wwise.
- The conversion settings that will be used to convert each file.

You must also specify the folder in which the converted sources will be saved so that they can be used by the Wwise sound engine at runtime.

To specify the input/output paths for external audio sources:

1. From the menu bar, click **Project > Project Settings**.

The Project Settings dialog box opens.

2. Switch to the **External Sources** tab.
3. In the Input Path group box, specify a folder where the External Sources List file is located by doing one of the following:

Click in the External Sources list and type the path where the External Sources List file is located.

Click the **Browse** button (...) that corresponds to one of the game platforms, navigate to the folder that contains the External Sources List file for that platform, and then click **Open**.

4. Repeat step 3 for each of the active platforms in your project.
5. In the Output Path group box, specify a folder where the converted external sources will be saved by doing one of the following:

Click in the External Sources Output Folder list and type the path where you want the audio files to be saved.

Click the **Browse** button (...) that corresponds to one of the game platforms, navigate to the folder where you want to save the converted audio files for that platform, and then click **OK**.

6. Repeat step 5 for each of the platforms.
7. Click **OK** to close the Project Settings dialog box.

Related Topics

- [Defining the General Settings for Your Project](#)
- [Defining the SoundBank Settings for Your Project](#)
- [Managing Messages that Appear in the Logs](#)
- [Defining Obstruction and Occlusion Curves for Your Project](#)
- [Enabling the Motion Devices for Your Project](#)
- [Defining the Default User Settings for your Project](#)

Specifying Network Ports

When you connect the Wwise authoring application to a game for profiling and real-time editing, communication between Wwise and the game is handled by a network using network ports that you open for the game and the authoring application. You can customize the ports that are opened and used by the authoring applications on the Network tab of the Wwise Project Settings dialog box. Ports opened / used by the game can be customized during initialization of the Wwise communication module by the game.

You can define a specific port number (fixed) or you can allow the operating system to automatically select one (dynamic). The choice is based on whether the port is opened by the authoring application or the game:



Caution

When changing this port in the project settings, be sure to change it to the exact same value in your game's code where the Wwise communication module is initialized (specifically the `ports.uDiscoveryBroadcast` member of the structure passed to `AK::Comm::Init()`).

Game Discovery Broadcast Port

This port is opened by the communication module within the game to listen to messages broadcast on the network by the authoring application when it tries to discover games it can connect to. The authoring application broadcasts Game Discovery messages to that port, so it is important to use the same port in the game and in your Wwise project. Since both the game and the authoring application need to know about this port, it cannot be dynamic (cannot be set to 0).



Caution

You have the option of using any port number between 1 and 65535, but verify that the port number you define does not conflict with any other applications running on the same computer or console as the game.

Game Discovery Response Port

This port is opened by the Wwise authoring application. This is where the game responds to the Game Discovery message. When defining the port number for this port, keep in mind the following important considerations:

- Setting this port to 0 will instruct the authoring application to request a dynamic port number, also known as an ephemeral port number. This means that instead of using a specific port number, the operating system will automatically select one. This is the default behavior and it is suggested to use a dynamic port number to avoid conflicts with other applications.
- If you need to use a fixed port number (that is, not a dynamic/ephemeral port number), avoid using the same port number as the Game Discovery Broadcast Port unless you do not plan on connecting to games running on the same computer as the authoring application.

- You have the option of using any port number between 0 and 65535, but verify that the port number you define does not conflict with any other applications running on the same computer or console as the game.

Customizing Communication Ports in the Game

As previously mentioned, the Game Discovery Broadcast Port is opened in the game by the communication module. There are three additional ports opened in the game by the communication module, and they are customized during initialization. Since these ports are not exposed in the authoring application, please refer to the "Initializing Communications" section of the [Wwise SDK documentation](#) for more information.

To customize communication ports:

1. Open the Project Settings dialog box by doing one of the following:
 - From the menu bar, click **Project > Project Settings**.
 - Press **Shift+K**.
2. Switch to the Network tab.
3. Edit port numbers as needed.
4. Click **OK**.

The Project Settings dialog box closes. The new port numbers will now be used for communication with games.

.

Related Topics

- [Connecting to a Local/Remote PC or Game Console](#)
- [Troubleshooting Remote Connection Issues](#)

Defining the Default User Settings for your Project

In Wwise, you can define a series of default user settings for a project in the Default User Settings dialog box. These values are a good starting point for defining properties in your project. It's a good idea to define these settings very early in your project development so they can be applied to all the objects you will create in your project.

Changing the default object settings in the middle of your project development will only affect objects created after the change. This feature can streamline your workflow because you can set up default properties for one type of object, such as SFX objects, and then modify the settings for another type of object, such as Voice objects.

These default settings are stored within the .wsettings file located in the current project directory. These settings can be defined by each user separately, which

allows users working on different areas of a project to set different default settings.

To define the default user settings for the objects in your project:

1. Open the Default User Settings dialog box by doing one of the following:

From the Project menu, select **Default Settings**.

Press **Shift+D**.

2. To set the default routing for the sound and music objects in your project, do the following:

In the Routing group box, click the **Browse** button (...) for the Audio output.

The Project Explorer - Browser opens.

Select the audio bus that you want to be the default bus for the sound and music objects in your project.

Click **OK**.

Any new sound or music objects will automatically be routed through this audio bus.

3. To set the default routing for the motion objects in your project, do the following:

In the Routing group box, click the **Browse** button (...) for the Motion output.

The Project Explorer - Browser opens.

Select the bus that you want to be the default motion bus for the motion FX objects in your project.

Click **OK**.

Any new motion FX objects will automatically be routed through this motion bus.

4. To define a default volume for all newly created objects, specify the value in the Volume text box.
5. To override the default conversion settings ShareSet used by the project, do the following:

In the Default Conversion Settings group box, select the **Override Project Settings** check box.

Click the **Browse** button (...).

The Project Explorer - Browser opens.

Navigate through the Conversion Settings hierarchy and select the ShareSet that you want to use as your project's default conversion settings.

Click **OK**.

All new top-level parent objects will automatically use the selected conversion settings ShareSet.

6. When you are finished, click **OK**.

The Default Settings dialog box closes and your default user settings are saved.

Related Topics

- [Defining your Project Settings](#)
- [Setting User Preferences](#)

Troubleshooting Your Project

You can check the status of your project by generating an integrity report that displays information about platform, audio file, SoundBank and plug-in issues. By examining this report, you can deal with project issues such as:

- Missing media files
- Missing audio or motion sources, such as missing language versions
- Plug-in problems
- Missing objects in SoundBanks
- Rendering and bypass effect issues for SoundBanks
- Rendering and RTPCs for SoundBanks

In addition to displaying the project issues, the integrity report also provides information about the issue including the following:

- **Platform** - The platform on which the issue occurs.
- **Type** - The type of object that is affected.
- **Object name** - The name of the object or element that is affected.
- **Status** - The description of the issue.
- **Comments/Suggestions** - Information and suggestions about how to deal with the issue.
- **Hierarchy** - The location of the affected object in the hierarchy.

Generating an Integrity Report

At any time during the development of your Wwise project, you can generate an integrity report. It is a good idea to generate the report prior to generating SoundBanks, or, if you are working with source control, before checking in major changes to a work unit.

To simplify looking for a specific type of issue, you can also filter the report by specifying the type of information that you want to generate. You can limit the report to display any or all of the following:

- Audio files and sources
- Hierarchies
- References
- Optimizations

After the report is generated, you can review each of the issues in the report and then double-click each one to carry out the suggested fix. For example, double-clicking a status message that states “No group assigned to the Switch container” opens the Property Editor where you can assign a switch or state group to the switch container.

To generate an integrity report:

1. From the Wwise menu bar, click **Views > Integrity Report**.

The Integrity Report opens.

2. To define which platform issues to include in the Integrity Report, select one or more of your project's platforms, as defined in the [Platform Manager](#), from the Platforms group box.
3. To define which project language issues to include in the Integrity Report, in the Languages group box, select the Languages option and choose the appropriate languages.
4. To filter for project issue types, select one or more of the following:

Audio files and sources

Hierarchies

Broken References

Optimizations

5. Click **Generate**.

A progress bar displays the status of the generation process.

6. When Wwise is finished generating the integrity report, click **Close**.

The Integrity Report- Completed dialog box closes and the Integrity Report is displayed.



Note

When you close the Integrity Report dialog box, the information is not saved. Make sure to save the report if you plan to view the information at another time. For information about how to save an integrity report, refer to [Saving an Integrity Report](#).

7. Double-click the issue to open the corresponding dialog box where you can fix or modify its issue status.

Integrity Report Issues

The following table lists the possible issues detected, and how you can correct these project issues.

Issue	Information/Suggestion
Original audio file not found.	Do one of the following: Drag the missing files into the Originals folder. Import the missing file using the Audio File Importer.
Audio source is missing.	Double-click the message to view the missing source in the Audio tab of the Project Explorer for SFX and in the Contents Editor for voice.
Source Plug-in not supported on platform.	Double-click the message to open the Contents Editor where you can select a source plug-in that is supported by the current platform.
Source Plug-in not installed.	Install the source plug-in.
Missing language sources.	Import the missing language source using the Audio File Importer.
Selected codec does not support this source's channel configuration.	Wwise will automatically downmix the source before conversion. Double-click the message and change the codec (audio format) or channels settings.
Audio FX plug-in not supported on platform.	Double-click the message to open the Property Editor where you can select an Effect plug-in that is supported by the current platform.
Audio FX Plug-in not installed.	Install the missing audio effects plug-in.
Render effect has been applied to an "In use" source that is not an audio file.	Render effects are not supported for source plug-ins. Double-click the message to do one of the following: In the Property Editor, remove the render effect option. In the Contents Editor, select an audio source that references an audio file to be "In Use".

Issue	Information/Suggestion
The selected effect must be rendered.	<p>Double-click to open the Property Editor and do one of the following:</p> <p>Render the effect.</p> <p>Change the effect used.</p>
The effect could be rendered to save CPU, since no parameters will change in game.	Double-click to open the Property Editor and select the Render check box.
The effect on this object has been bypassed. The render effect will be ignored.	Since both of these operations cannot be supported, verify the importance to your project of the render effect. If you want to apply the render effect option, remove the Bypass Effect.
A bypass effect property has been assigned to the RTPC for this object. The render effect will be ignored.	Since both of these operations cannot be supported, verify the importance to your project of the Render effect operation or the RTPC-driven bypass effect. If you want to apply the render effect option, remove the RTPC bypass effect on the object.
An RTPC has been applied to the effect for this object. The render effect will be ignored.	Since both of these operations cannot be supported, verify the importance to your project of the RTPC that has been applied, or the render effect. If you want to apply the render effect, remove the RTPC.
Streamed XMA files do not support region loops. All user-defined loop regions will be ignored.	<p>Do one of the following:</p> <p>Double-click the message to open the Property Editor and clear the Loop check box.</p> <p>In the Conversion Setting dialog box, change the audio format of the file.</p>
XMA files cannot have a loop setting that exceeds 254.	<p>Do one of the following:</p> <p>Double-click the message to open the Property Editor and in the No. of loops text box, reduce the number of loops.</p> <p>In the Conversion Setting dialog box, change the audio format of the file.</p>
In memory XMA files of over 8 Mb are not supported.	<p>Do one of the following:</p> <p>Stream the sounds.</p> <p>Reduce the sample rate or the compression quality.</p>
Seek table required for this virtual voice behavior.	<p>Double-click the message to open the Conversion Settings dialog box and do one of the following:</p> <p>Enable the seek table in the Vorbis Encoder Parameters dialog box.</p> <p>Change the file format.</p>
Seek table not required for this virtual voice behavior.	To reduce the memory usage, you can disable the seek table. Double-click the message to open the Conversion Settings dialog box to disable the seek table in the Vorbis Encoder Parameters dialog box.
Seek table required for music objects in Vorbis format.	Double-click the message to open the Conversion Settings dialog box and do one of the following:

Issue	Information/Suggestion
	<p>Enable the seek table in the Vorbis Encoder Parameters dialog box.</p> <p>Change the file format.</p>
<p>Child object in this container is not assigned to a switch.</p>	<p>Double-click the message to open the Project Explorer and the associated Property Editor where you can do one of the following:</p> <p>Remove the unassigned object.</p> <p>Assign the object to a switch.</p>
<p>No group assigned to the Switch container.</p>	<p>Double-click to open the Property Editor and assign a switch or state group to the switch container.</p>
<p>No objects assigned to a switch in Switch container.</p>	<p>Double click to open the Contents Editor and assign an object to a switch in the switch container.</p>
<p>No group is assigned to the Music Switch container.</p>	<p>Double-click to open the Property Editor and assign a switch or state group to the music switch container.</p>
<p>No music objects assigned to a switch in this container.</p>	<p>Double-click to open the Music Switch Association Editor where you can assign music objects to the switch.</p>
<p>The playlist of the Sequence container has no assigned objects.</p>	<p>Double-click to open the Contents Editor where you can assign objects to the playlist.</p>
<p>The playlist of the Music Playlist container has no assigned objects.</p>	<p>Double-click to open the Music Playlist Editor and assign music objects to the playlist.</p>
<p>A blend track in the blend container has no assigned objects.</p>	<p>Double-click to open the Blend Track Editor where you can do one of the following:</p> <p>Add objects to the blend track.</p> <p>Remove the blend track.</p>
<p>A music track in the Music segment contains no audio files.</p>	<p>Double-click to open the Music Segment Editor where you can do one of the following:</p> <p>Add clips to the music track.</p> <p>Delete the music track.</p>
<p>The SoundBank refers to objects that no longer exist.</p>	<p>Delete the events or objects from the SoundBank.</p>
<p>No object assigned to event action.</p>	<p>Double-click and open the Event Editor to do one of the following:</p> <p>Assign objects to these actions.</p> <p>Remove the event action.</p>
<p>No object assigned to path in dialogue event.</p>	<p>Double-click to open the Dialogue Event Editor and do one of the following:</p> <p>Assign an object to the path.</p> <p>Remove the path.</p>
<p>Some child objects in this sample accurate container are not using the same</p>	<p>The parent container may no longer be sample accurate. Double-click to open the Schematic View where you can</p>

Issue	Information/Suggestion
output properties as the parent container.	remove any output overrides for the child objects of the sample accurate container.
Some child objects in this sample accurate container are not using the same effect properties as the parent container.	The parent container may no longer be sample accurate. Double-click to open the Schematic View where you can remove any effect overrides for the child objects of the sample accurate container.
Some child objects in this sample accurate container are not using the same advanced settings properties as the parent container.	The parent container may no longer be sample accurate. Double-click to open the Schematic View where you can remove any advanced settings overrides for the child objects of the sample accurate container.
Some containers in this sample accurate container are not sample accurate.	The parent container may no longer be sample accurate. Double-click to open the Project Explorer and change the non-sample accurate child containers of the sample accurate container to sample accurate.
This sample accurate container has a child switch container.	The parent container may no longer be sample accurate. Double-click to open the Project Explorer to remove the switch container.
An audio source in this sample accurate container does not have the same number of channels as other audio sources in the container.	The parent container may no longer be sample accurate. Verify that all audio sources for this container have the same number of channels.
This sample accurate container contains both sound and motion objects, which is not supported.	Remove the Motion FX object from the container.
Some audio files are shorter than 0.05 seconds. These cannot be used for crossfade transitions.	<p>Double-click to open the Project Explorer to locate the sound object associated with the audio file whose duration is too short for a crossfade transition and do one of the following.</p> <p>Drag and drop another audio file with a longer duration onto the sound object.</p> <p>In the Property Editor for the parent object, select another transition type.</p>
Switch containers with more than one object assigned to a switch do not crossfade correctly.	Double-click to open the Project Explorer and in the associated Property Editor ensure that only one object is assigned to a switch for crossfade transitions.
A container with a trigger rate duration of less than 0.021 seconds may not play as expected.	Double-click to open the Property Editor and in the associated Property Editor, change the trigger rate duration to more than 0.021 seconds.
This Motion FX object is not routed to a Motion Bus.	Specify a motion output bus for this object or one of its parent containers.
This source only contains an LFE channel.	The LFE channel is discarded on the Wii.
This source contains an LFE channel.	The LFE channel is discarded on the Wii.

Saving an Integrity Report

It is a good idea to save the integrity report so that you can consult the report to resolve issues, or to keep a record of the issues in your project.

To save an integrity report:

1. In the Integrity Report, click **Save Report**.

The Save As dialog box opens.

2. Browse to the folder where you want to save the report.
3. Replace the default report name with one that best represents this report, and click **Save**.

The integrity report is saved as a text file in the location that you have specified.

The Learning Annex - More on Obstruction and Occlusion

Obstruction and occlusion are conditions that exist when the space between a sound source and a listener is blocked. Obstruction refers to a partial block, and occlusion to a complete block. These conditions are set up by the game developers, who programmatically define the geometry of the game. Wwise does not compute obstruction and occlusion levels by itself. The physics calculations must be done by the game and the results must then be passed to the sound engine using the following function:

```
SetObjectObstructionAndOcclusion()
```

In Wwise, you simply define the volume and LPF applied to sound objects as they are influenced by obstruction and occlusion during gameplay. The values you define are relative, and are added to any volume and LPF values already applied to the game object.

In the implementation of these conditions, percentage values are assigned to both obstruction and occlusion to enhance realism. These percentages are based on real-time positioning information from the game engine, and can reflect a game object's movement in and out of a blocked area. For example, a spy character might move through an art gallery full of large sculptures. The percentage of obstruction could change depending on where she is as she passes behind each sculpture. Similarly, the percentage of occlusion could shift as the spy slowly sneaks out of a completely occluded broom closet.



Note

The percentage value calculations for obstruction and occlusion are handled by the sound engine. You do not specify these values in Wwise.

For a better understanding of when obstruction and occlusion occurs, refer to the following topics:

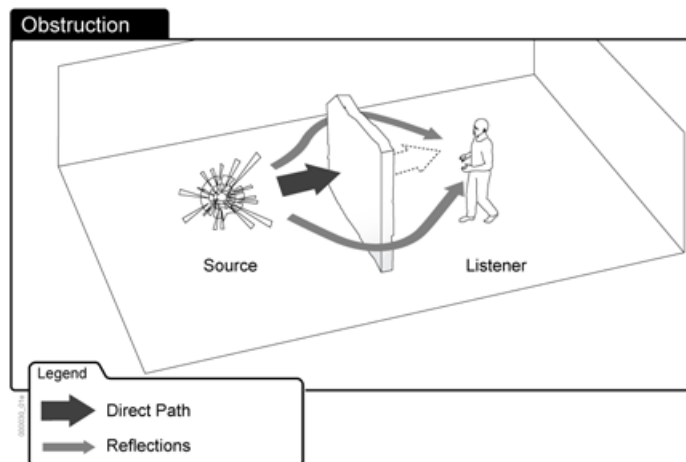
- [Understanding Obstruction](#)
- [Understanding Occlusion](#)

Understanding Obstruction

Obstruction occurs when an object in the game geometry, such as a wall or pillar, partially blocks the space between a sound source and a listener. For example, in a spy game, player characters could hear gunshots from ahead of them despite ducking behind a pillar.

When obstruction occurs, the listener hears the reflections of the sound clearly, but the direct path of the sound is obscured. This alteration to the sound is performed through the application of a reduction in volume, an increase in LPF, or both.

The following illustration demonstrates the condition of obstruction.

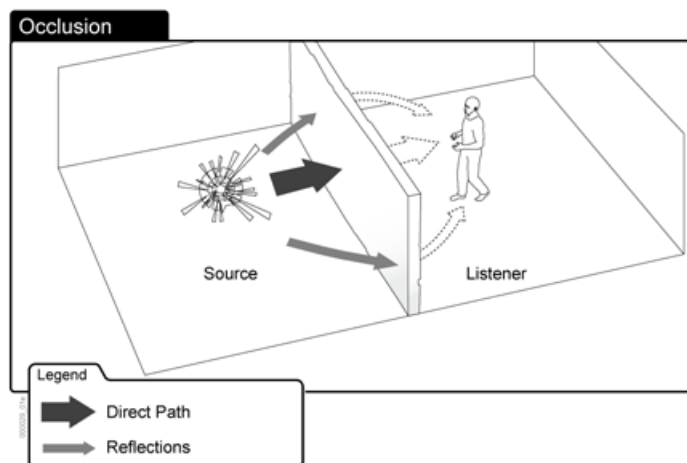


Understanding Occlusion

Occlusion occurs when an object in the game geometry completely blocks the space between a sound source and its listener. For example, in a spy game, the player character might hear gunshots in the next room, despite there being a wall between the character and the discharged firearm.

When occlusion occurs, the direct and reflected paths of the sound are both obscured. As with obstruction, you can alter both the volume and the LPF of the sound.

The following illustration demonstrates the condition of occlusion.



Note

Obstruction and occlusion settings can be used in parallel with an Environmental effect to change sound properties dynamically depending on a source's location. For more information on using Environmental effects, refer to [Understanding Sends](#).

Project Management Tips and Best Practices

You may want to review the following sections, which provide you with a series of tips and best practices that can help you better manage your project.

Using the Integrity Report

It is a good idea to generate an Integrity Report regularly to handle any project issues, but particularly in the following situations:

- **Before generating SoundBanks** - So that you can address project issues before packaging your audio and motion into a SoundBank.
- **Before checking in major changes to a work unit** - So that you can ensure that your changes have not created project issues.

Using Default Settings

You can use your default settings to “multi-edit” the default routing and volume for your project. When you change any of these values, any objects added

subsequent to the change will have the new default values. So you can have different default values based on objects added after any change that you make to these settings.

Defining Obstruction and Occlusion Curves

It is a good practice to always define your curves in a linear fashion to minimize CPU and memory usage in your project. Keep your curves as simple as possible to begin with and customize only as needed.

Dividing Large Projects into Work Units

If you are several people working on a large project, you may want to divide up your project into smaller pieces, using work units. Work units are distinct XML files that contain information related to a particular section or element within your project. These work units can help you organize and manage the different elements within a project. If you are working as part of a team, these work units can also be managed by your source control system to make it easier for the different members of your team to work on the project concurrently. After your project is divided into work units, you can unload some of these work units, keeping only those that you are working on. The main reason for unloading work units from a project is to speed up project load times, reduce memory usage, and improve overall performance in Wwise. If your project is very large, you can improve performance significantly by unloading one or more work units.

Chapter 4. Managing Platforms

Overview	101
Platform Manager	101
PS Vita Hardware Versus Software Platforms	104

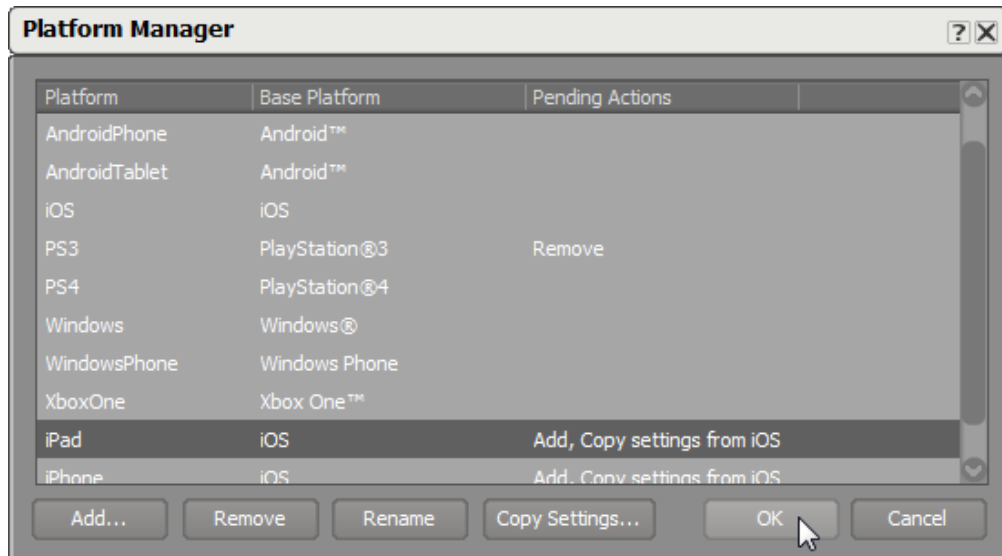
Overview

A game is often released for multiple platforms. Consequently, it is essential to be able to specify the target platforms for a project. In Wwise, this is done via the Platform Manager.

In the Wwise menu bar, select **Project > Platform Manager...** to prompt the [Platform Manager](#). Default Shortcut: Shift+Alt+P.

Platform Manager

The Platform Manager, seen below, allows you to define the platforms for your game. A platform consists of a platform name and a base SDK. You can specify as many platforms as desired for any listed SDK.



The Platform Manager

Wwise includes a list of base platforms which cover the primary SDKs used with Wwise, and you are free to create your platforms according to all the base platforms pertinent to your project. But, additionally, you may wish to add multiple platforms with the same base platform in order to accommodate differences.

For example, you may have both iPhone and iPad platforms to develop in your project. They are not exactly the same even though they are fundamentally iOS platforms. To facilitate your management of these distinct project types, you could create two iOS-based platforms: one specific to iPad and another specific to iPhone.

Related Topics

- [Adding, Removing, and Copying Platforms](#)

Adding, Removing, and Copying Platforms

1. In the Wwise menu bar, select **Project > Platform Manager...**

The Platform Manager is displayed.

Adding a Platform

1. Click **Add...**

The **Add Platform** dialog box is displayed.



Note

If a platform is highlighted in the **Platform Manager** when clicking **Add Platform...**, then all three fields are populated with that platform's default information. Otherwise, the fields are populated with information from the alphabetically first listed platform, **Android™**.

2. Select a **Base Platform** from the list of SDKs.

The name field updates accordingly.

3. If needed, select **Override Default Name**. This activates the name field, where you can specify the new platform's name. It must be alphanumeric (no spaces, but underscores are accepted) and unique among all defined platform names.



Tip

If possible, avoid overriding the default name because third-party Wwise plug-ins may reference it.

4. Optionally, select an existing platform to **Copy settings from**.
5. Click **OK** to confirm the new platform information and close the **Add Platform** dialog.

The new platform name, base platform, and pending actions ("Add" and possibly "Copy settings from...") appear in their respective columns of the new bottom row of the **Add Platform** dialog.



Note

If desired, you can click **Remove** to instantly delete a selected new entry.

6. Click **OK** to commit the newly added platforms.

A warning appears explaining that these platform changes cannot be undone.

7. Click **Yes** to proceed with the pending actions.

Another warning appears to remind you to verify the platform changes, especially in path names, after the project reloads.

8. Click **OK** to reload the project.

The project closes and reloads with the new platform settings. Depending on the size of your project, this may take a few seconds.

Removing a Platform

1. Select the listed platform you want to remove.
2. Click **Remove**.
3. The **Pending Actions** column of the selected platform is marked **Remove**.



Note

To cancel the **Remove** action, you can close the **Platform Manager** or click **Cancel**.

4. Click **OK** to activate the removal of the selected platforms.

A warning appears explaining that these platform changes cannot be undone.

5. Click **Yes** to proceed with the pending actions.

Another warning appears to remind you to verify the platform changes, especially in path names, after the project reloads.

6. Click **OK** to reload the project.

The project closes and reloads with the new platform settings. Depending on the size of your project, this may take a few seconds.

Copying Platform Settings

You may select to copy the platform-specific settings from a previously defined platform over to another platform.

This could save you a lot of time if many of the platforms' settings should be the same.

1. Click **Copy settings**....

This prompts the **Copy Platform Settings** dialog box.

2. Select the source platform in the **From** list.
3. Select the destination platform in the **To** list.
4. Click **OK** to commit the newly added platforms.

A warning appears explaining that these platform changes cannot be undone.

5. Click **Yes** to proceed with the pending actions.

Another warning appears to remind you to verify the platform changes, especially in path names, after the project reloads.

6. Click **OK** to reload the project.

The project closes and reloads with the new **Platform-specific settings that are copied over from the source to the target platform**. Depending on the size of your project, this may take a few seconds.



Caution

Be sure to have an up-to-date backup of your project in case you decide to revert these changes later.



Warning

This operation cannot be undone, and it will clear the Wwise-related contents on your clipboard and your Undo history.

Related Topics

- [Copying Settings from One Platform to Another](#)

PS Vita Hardware Versus Software Platforms

Wwise has two distinct platform bases for PS Vita. They are based on different approaches for mixing with PS Vita:

- PS Vita HW - The Wwise hardware mixing pipeline uses a Vita-specific chip that handles sound compression decoding and effects.

- PS Vita SW - The Wwise software mixing pipeline works like it does with all other platforms; it does not support the Vita-specific chip.

The Wwise software and hardware mixing pipelines for PS Vita each have their advantages and disadvantages. To help in making an informed decision, the following points should be taken into consideration.

CPU Usage

The number of active voices and effect plug-ins are the two major culprits for CPU usage. Reducing them will save on your CPU, so the following factors must be considered.

- **Codec Decompression** - Using [ATRAC9](#) or [VAG](#) on Vita presents different issues for the chosen Wwise mixing pipeline. [VAG](#) decompression is only available through the hardware mixing pipeline, while [ATRAC9](#) is available both through the Wwise software and hardware mixing pipelines. However, Wwise's hardware mixing version uses [ATRAC9](#) more efficiently; the software mixing version has a larger overhead and a limited voice count.
- **Wwise Vita Effect Plugins** - In the software mixing pipeline, all the standard Wwise plug-ins are available, thereby offering numerous effects to designers. In the hardware mixing pipeline, however, Wwise only has five Vita effect plug-ins. This results in savings for CPU usage, although it also limits designers.



Note

Wwise Vita effect plug-ins were developed by Sony for use on the PS Vita. Learn more about these in the Reference section of the User Help and, for greater detail, please visit the Sony Developer Network for Vita.

Porting

When porting projects to or from Vita, such as from PS4 to PS Vita or PS Vita to iOS, the software mixing pipeline keeps most if not all of your work. However, the hardware mixing pipeline, which has a different set of effects, will require you to redo a lot of your work.

Wwise Hardware Mixing Pipeline Limitations

In addition to supporting only a limited number of effects, the Wwise hardware mixing pipeline does not support *Sample accurate* transitions for [ATRAC9](#) or [VAG](#). The *Interactive Music Engine* is dependent on using either *Sample accurate* transitions or *Sample accurate* seeking; but, *Sample accurate* seeking is not supported for either the software or hardware mixing pipeline on [ATRAC9](#)

and [VAG](#). So, in practical terms, the *Interactive Music Engine* is only available through the software mixing pipeline.



Note

Sample accurate transitions and the **Interactive Music** engine are both fully supported through other codecs like [PCM](#) and [Vorbis](#).

Conclusion

In summary, the Wwise hardware pipeline for PS Vita can save you on CPU usage. However, it could creatively limit your designers and largely remove the savings in time and effort gained through the standard software pipeline's capacity to easily port a project from one platform to another. In deciding on which approach is best, you have to weigh the relative importance of these elements for each project.

Chapter 5. Managing Workgroups

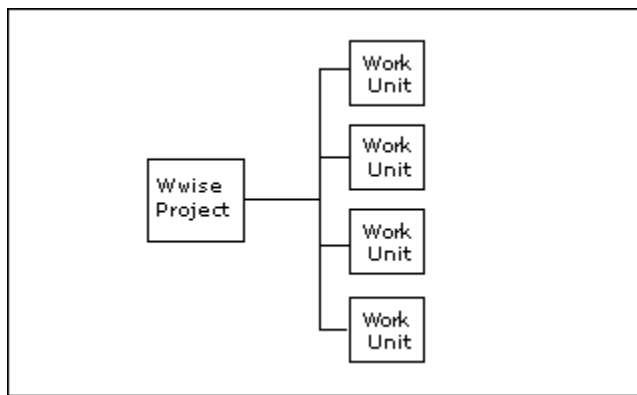
Overview	108
Dividing Your Project into Work Units	111
Viewing the Status of Your Project Files	120
Using Wwise with Your Source Control System	122
Resolving Project Inconsistencies	123
Managing Project Files Using a Workgroup Plug-in	125
Workgroup Tips & Best Practices	138

Overview

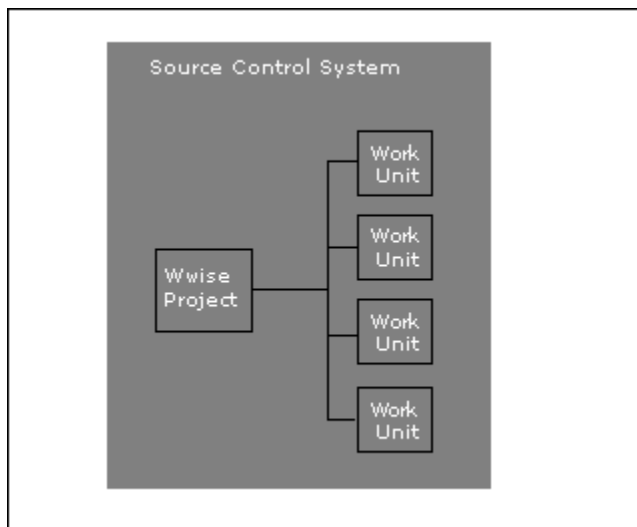
In today's game environment, with the complexity of games and the pressure to get games to market, it is essential for sound designers, music composers, audio integrators, and audio programmers to be able to work together on the same project. In Wwise, you can do this using Workgroups.

You must use only one Wwise project per game.

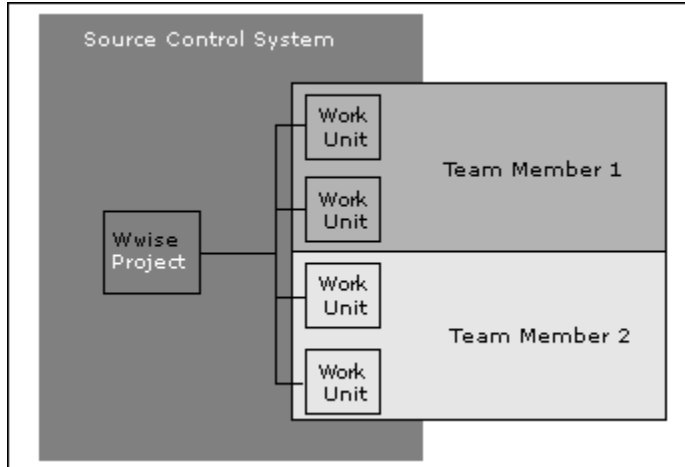
In order for several people to work efficiently on the same project, it must be broken up into smaller pieces. In Wwise, these pieces are called work units.



These work unit files can then be managed by your source control system.

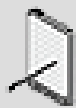


Each person within your workgroup can then work on the same or different parts of the project in parallel.



In most cases, you will want different people working on different parts of the project in order to avoid difficult and frequent merging issues. However, there will be cases when two or more people will have to work on the same work unit concurrently. When these files are checked back in to your source control system, you will most likely have to deal with merge conflicts. Refer to your source control documentation for more information on how best to deal with merge conflicts.

Although Wwise is not a source control management system, you can use its open architecture to easily integrate your existing source control system. This allows you to manage your project assets and perform many of your source control functions directly in Wwise. For more information on the Wwise source control plug-in, refer to [Managing Project Files Using a Workgroup Plug-in](#).



Note

In order to create a workgroup plug-in, your source control system needs to support third-party integration using their API. Perforce and Subversion plug-ins are installed with Wwise.

What are Work Units?

At the foundation of Wwise Workgroups is the work unit. Work units are distinct XML files that contain information related to a particular section or element within your project. These work units can help you organize and manage the different elements within a project. If you are working as part of a team, these work units can also be managed by your source control system to make it easier for the different members of your team to work on the project concurrently.

When a project is created, a default work unit is created for each of the following elements in Wwise:

- Actor-Mixer hierarchy
- Attenuations
- Control Surface Session
- Conversion Settings
- Dynamic Dialogue
- Effects
- Events
- Game Parameters
- Interactive Music hierarchy
- Master-Mixer hierarchy
- Mixing Sessions
- Modulators
- Presets
- Queries
- SoundBanks
- Soundcaster Sessions
- States
- Switches
- Triggers

These default work units are located in their respective folders within your project directory. Each one is named “Default Work Unit.wwu”. The default work units are created so that you can begin creating objects, events, states, and so on, without having to create work units for each project element first.

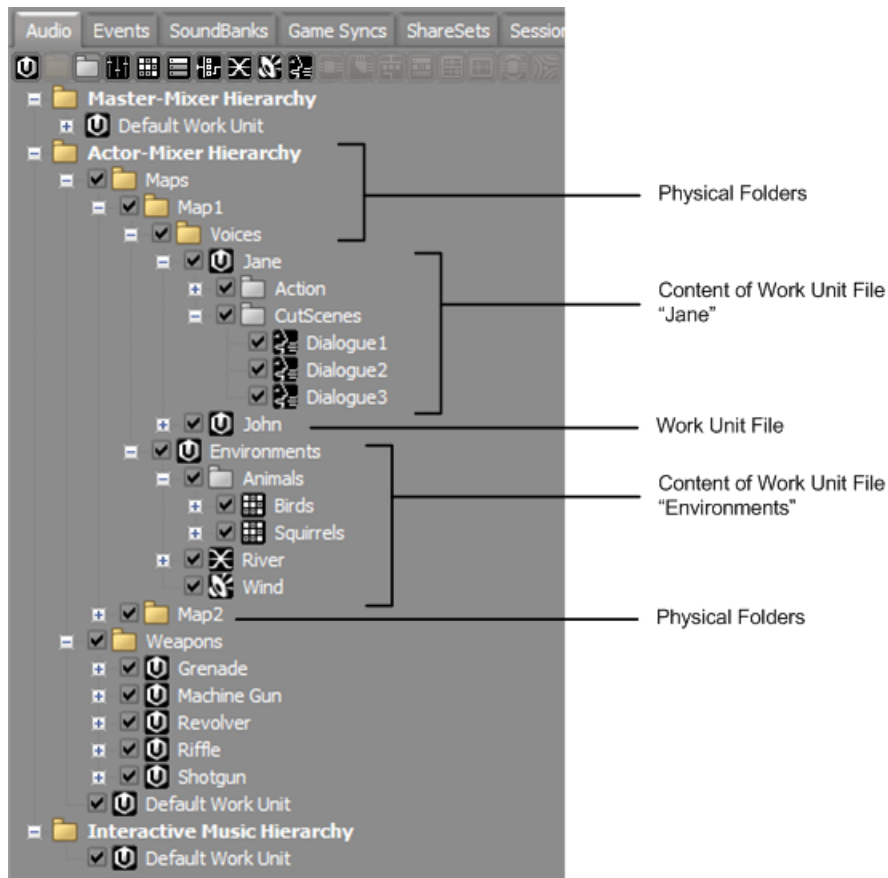
As your project grows or if more people join the project team, you may want to divide up the different project elements into new work units. For example, you can create three different work units for states called StatesLevel1, StatesLevel2, and StatesLevel3.

If you decide to create new work units, you can leave the default work units empty. The default work units, however, are critical project files and should not be renamed or deleted. If you rename or delete these files, Wwise will recreate them the next time you open the project.

Work units can be organized into physical sub folders on disk. The physical folder structure is replicated in the Project Explorer.

You can create new work units for all project elements except Presets and the Master-Mixer Hierarchy. The busses in the Master-Mixer Hierarchy must always remain in their default work unit.

The following example shows how work units can be created and organized to divide up the sound structures in the Actor-Mixer Hierarchy.



You can create and manage the contents of your work units in the different tabs of the Project Explorer. For more information about organizing your project into work units, refer to [Dividing Your Project into Work Units](#).

Dividing Your Project into Work Units

Before dividing up your project, you must first decide on the best way to organize the different elements in your project to streamline your workflow and minimize conflicts. Because there are many ways to divide up a project, it's worth spending the time to decide on the best way for your project.

After the work units are created, you can divide up the work in your project by dragging the sound structures, events, SoundBanks, and so on into their respective work units. You can also unload specific work units from your version of the project, keeping only those that you are working on, to reduce project load times and memory usage.

Managing the work units within your project consists of the following tasks:

- [Creating Work Units in Your Project](#)
- [Assigning Project Elements to Work Units](#)
- [Loading/Unloading Work Units from Your Project](#)

Creating Work Units in Your Project

You can create new work units for the following elements in your project:

- Sound structures in the Actor-Mixer hierarchy
- Music structures in the Interactive Music hierarchy
- Action events and dialogue events
- SoundBanks
- Switches, states, game parameters, and triggers
- Effect and attenuation shareSets
- Soundcaster and Mixing Desk sessions
- Queries

Each work unit can only contain those project elements for which it was created. For example, an event work unit can only contain events, an effects work unit can only contain effects, and so on.

If you plan to use a source control system to manage your project files, you can add all the work units, including the default work units, to the central repository or depot of your source control system. If you create a work unit while using a workgroup plug-in, you will be prompted to add the file to the repository.

To create a work unit:

1. In the Project Explorer, right-click one of the Physical Folders (For example: **Actor-Mixer Hierarchy** or **Interactive Music Hierarchy**).

A shortcut menu is displayed.

2. Click **New Child > Work Unit**.

The New Work Unit dialog box opens.

3. In the Name field, type the name of the work unit.



Note

The following characters may not be used when naming work units in Wwise: ‘:<>*?"/\|.’

4. Click OK.

A new work unit is created.



Note

You can delete or rename this work unit directly in the Project Explorer or in the File Manager. To open the File Manager, click Projects > File Manager.

Related Topics

- [Assigning Project Elements to Work Units](#)
- [Renaming User-Created Work Units](#)
- [Deleting User-Created Work Units](#)
- [Workgroup Tips & Best Practices](#)

Assigning Project Elements to Work Units

By default, all the elements in your project, including sound structures, events, and SoundBanks, are automatically assigned to their respective default work units. After creating new work units, you can assign the different elements in your project to the newly created work units. You can assign a project element to a work unit by simply dragging it into a particular work unit.



Note

Before moving project elements to a new work unit or between existing work units, you should make sure that other people within your project team aren't currently working on the affected project elements.

To assign project elements to work units:

1. In the Project Explorer, click one of the following tabs:

Audio

Events

SoundBanks

Game Syncs

ShareSets

Sessions

Queries

2. Drag one or more project elements from the default work unit to one of the work units you created.

The project element is now assigned to the new work unit.

Related Topics

- [Creating Work Units in Your Project](#)
- [Workgroup Tips & Best Practices](#)
- [Renaming User-Created Work Units](#)

Managing Physical Folders

Work units can be organized into Physical Folders on disk inside the root folder associated to a specific category of objects (example: *Actor-Mixer Hierarchy*). This is especially useful when you have hundreds of work units and don't want to have them flat in the same folder.

Physical Folder and Virtual Folder are two very distinct entities. The physical folder represents an actual directory on disk, while the virtual folder is part of its containing work unit file. Both are used to organize content, but the physical folder is an ancestor of a work unit and the virtual folder is a descendant of a work unit.

Physical folders can be created directly inside the Project Explorer by using the contextual menu on other physical folder, or by using the Project Explorer toolbar button.

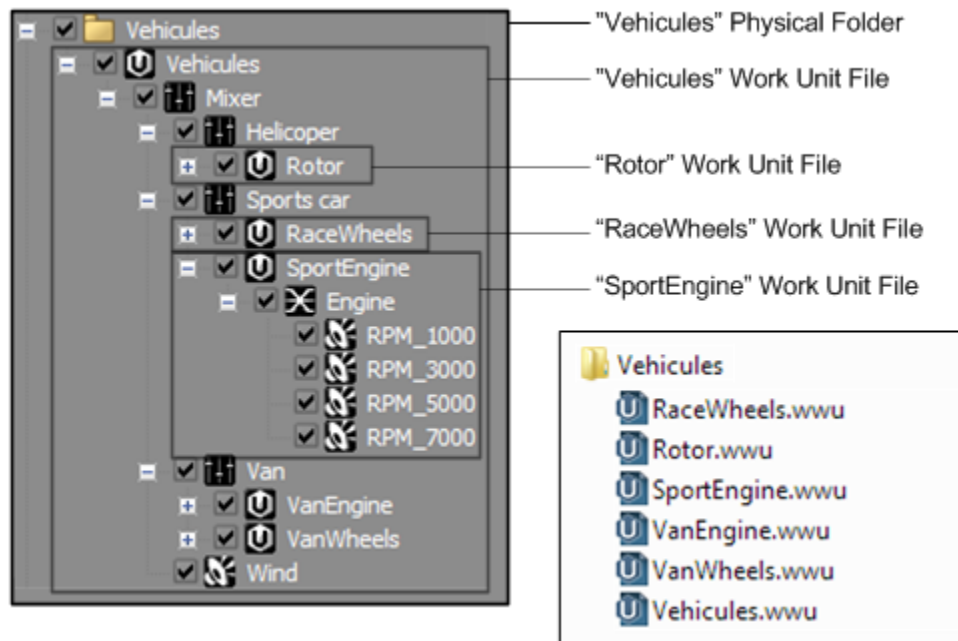
Physical folder can only contain other physical folders or work units files.

Nested Work Units

Work units can be nested inside other work units. When this is done, the content of the work unit is delegated to the nested work unit, instead of the parent work unit. This allows to place several work units under a common actor-mixer object, providing common behavior and mixing properties for the nested work units.

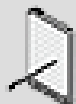
The nested work units allows a finer granularity in the file separation, reducing potential conflicts when merging.

Nested work units are saved in their containing physical folder. For work units, names must be unique across a single physical folder.



Nested work units can be created under the following object types:

- Physical folders
- Work units
- Virtual folders
- Actor-Mixer



Note

Nested Work Units are stored on disk at the same level as their root Work Unit.

Loading/Unloading Work Units from Your Project

After your project has been divided up into work units, you may want to unload some of these work units, keeping only those that you are working on. The main reason for unloading work units from a project is to speed up project load times, reduce memory usage, and improve overall performance. If your project is very large, you can improve performance significantly by unloading one or more work units.

The following types of user-created work units can be unloaded from a project:

- Actor-Mixer Hierarchy work units
- Interactive Music Hierarchy work units

- Event work units



Note

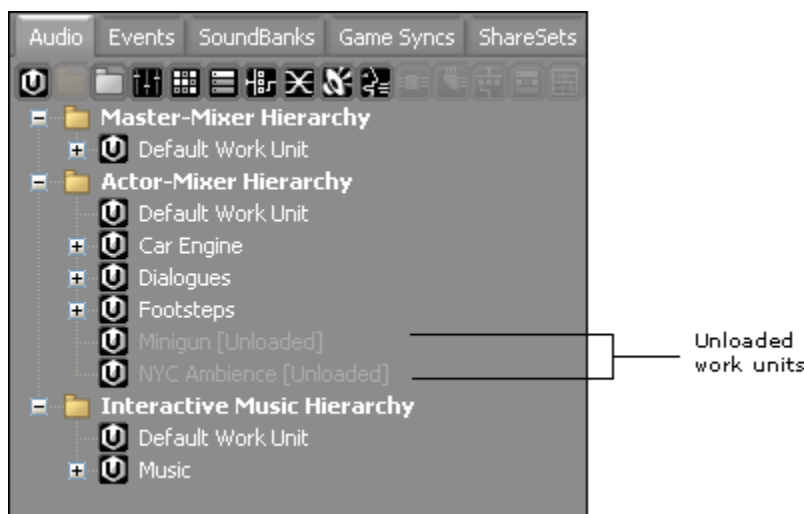
Default work units are critical to the project, so they can never be unloaded.



Note

Nested work units can't be unloaded individually, but they will be unloaded if you unload their root work unit.

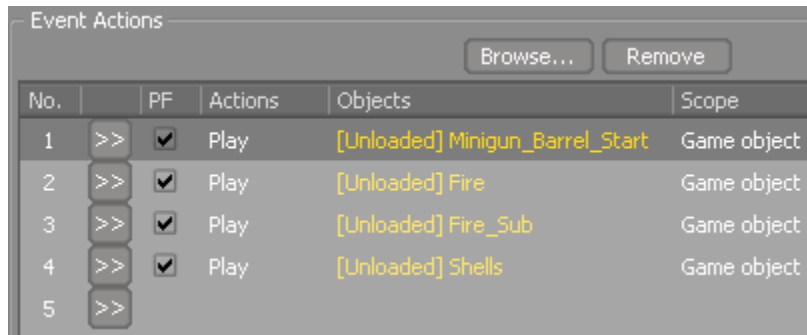
When work units are unloaded from a project, they will appear “ghosted” in the Project Explorer. You will not be able to work with them until they have been loaded back into the project.



You can continue to work with the objects and events within the loaded work units. You may, however, discover that some of these objects and events contain references to objects and/or events that have been unloaded. For example, you may have references to unloaded objects or events in the following project elements:

- Dialogue events
- Events
- Music stingers
- Music transitions
- Queries
- SoundBanks
- Soundcaster and mixing sessions

In these cases, the unloaded elements will be marked as [Unloaded] and will be highlighted in yellow.



No.	PF	Actions	Objects	Scope
1	>>	<input checked="" type="checkbox"/> Play	[Unloaded] Minigun_Barrel_Start	Game object
2	>>	<input checked="" type="checkbox"/> Play	[Unloaded] Fire	Game object
3	>>	<input checked="" type="checkbox"/> Play	[Unloaded] Fire_Sub	Game object
4	>>	<input checked="" type="checkbox"/> Play	[Unloaded] Shells	Game object
5	>>			

You can still add/remove objects, modify property values, generate SoundBanks, and so on, but you won't be able to play back these objects and events in Wwise.

When you unload a work unit from a project, it doesn't get unloaded immediately. The work units you have selected for “unloading” are only unloaded from a project when the project is closed. The next time you open the project, Wwise will not load the work units that have been unloaded. If you need to access the contents of an unloaded work unit, you can easily load it back into the project at any time. If any changes are made to the project that affects the objects or events within the unloaded work unit, these changes will be listed in the Project Load Log when the work units are loaded back into the project.



Note

When work units are unloaded from a project, Wwise can't distinguish between source files that are not being used and those that haven't been loaded into the project. As a result, all source files that don't appear to be used by the project will be marked as “Unknown” in the Usage column of the File Manager until all work units are loaded back into the project and Wwise can perform a complete verification.

To unload a work unit from your project:

1. In the Project Explorer, select the work units that you want to unload.
2. Right-click the selection and select **Unload Work Unit (at next project load)** from the menu.
3. Save your project by pressing **Ctrl+S**.
4. Re-open the project by clicking **Project > Project Name**.

The work units you previously unloaded are no longer loaded when the project opens.

To load a work unit back into your project:

1. In the Project Explorer, select the unloaded work units that you want to load back into the project.
2. Right-click the selection and select **Load Work Unit** from the menu.

The work units are loaded back into the project.



Note

If any changes were made to the project that affected the objects or events within the unloaded work unit, these changes will be listed in the Project Load Log when the work units are loaded back into the project.

Related Topics

- [Creating Work Units in Your Project](#)
- [Assigning Project Elements to Work Units](#)
- [Renaming User-Created Work Units](#)
- [Deleting User-Created Work Units](#)

Renaming User-Created Work Units

At some point you may need to rename a work unit that you have created in your project. You can rename work units in the **Project Explorer** and you can rename them in the **File Manager**. Work units should not be renamed or deleted in Windows Explorer or Mac Finder because you may cause integrity errors or lose project data.



Caution

The default work units are critical project files and should not be renamed or deleted. If you do, Wwise will automatically re-create them the next time you open the project.

To rename a work unit:

1. Do one of the following:

From the **Project Explorer**, press *F2* or use the **Rename** shortcut menu option.

The editable text box appears over the work unit name.

2. In the edit box, type the new name and press *Enter*.

The *Workgroup Operation* dialog appears.

3. Verify the operation to be applied and click **Continue**.

If you are using a source control plug-in, verify if the operation was successful.

4. Click **OK** to close the Process Log.

Related Topics

- [What are Work Units?](#)
- [Creating Work Units in Your Project](#)
- [Assigning Project Elements to Work Units](#)
- [Deleting User-Created Work Units](#)

Deleting User-Created Work Units

You may need to delete a work unit that you have created and no longer need in your project. You can delete work units in the **Project Explorer** and you can delete them in the **File Manager**. Work units should not be deleted in Windows Explorer or Mac Finder because you may cause integrity errors or lose project data.

If you delete a work unit file from your source control system, the project elements within that work unit will no longer exist in the project, which can cause integrity errors. These errors are identified when you open the project along with a description of how they will be fixed, if necessary. For more information about project inconsistencies, refer to [Resolving Project Inconsistencies](#).



Caution

The default work units are critical project files and should not be renamed or deleted. If you do, Wwise will automatically re-create them the next time you open the project.

To delete a work unit:

1. Do one of the following:

From the Project Explorer, select the work unit to delete and press *Delete* or use the *Delete* function in the shortcut menu.

The *Workgroup Operation* dialog appears.

2. Verify the operation to be applied and click **Continue**.

If you are using a source control plug-in, verify if the operation was successful.

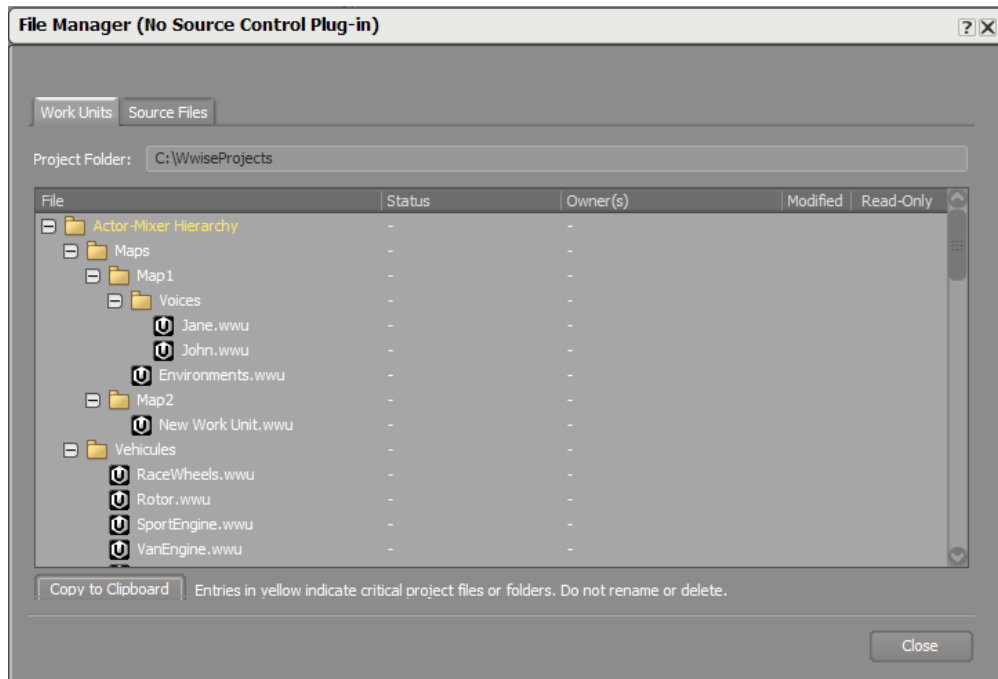
3. Click OK to close the Process Log.

Related Topics

- [What are Work Units?](#)
- [Creating Work Units in Your Project](#)
- [Assigning Project Elements to Work Units](#)
- [Renaming User-Created Work Units](#)

Viewing the Status of Your Project Files

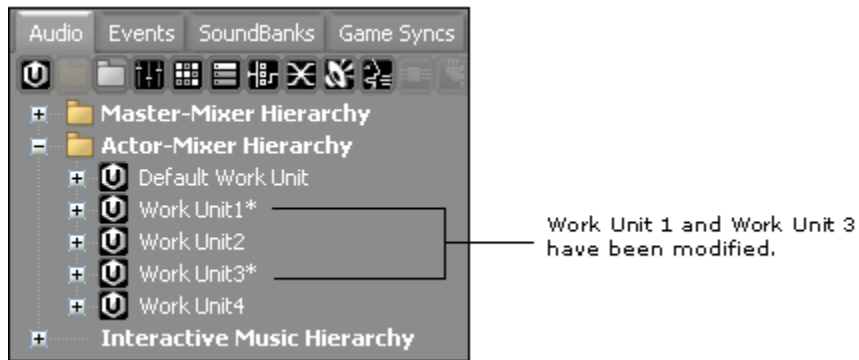
As your project grows, you will need to keep track of the various files within your project. The File Manager displays information about each audio file and work unit in your project, as well as the project file itself. You can use the File Manager to verify the status and owner of a project or source file as well as to determine whether a source audio file is being used by audio sources in your project.



When a work unit has been modified or has been tagged as read-only, a check mark appears in the corresponding column for that project file. This kind of feedback can help you keep track of which files you have modified and which ones cannot be modified.

The File Manager also displays information about which files are being used by the project so that you can regularly clean up files that are no longer in use. When work units are unloaded from a project, however, Wwise can no longer distinguish between source files that are not being used and those that haven't been loaded into the project. In these situations, all source files that don't appear to be used in the project will be marked as “Unknown” in the Usage column, until Wwise can perform a complete verification.

Wwise also provides feedback on which files have been modified directly in the Project Explorer. Just like with projects, Wwise uses an asterisk (*) to show which work units have been modified and require saving.



Caution

If you make changes to work units that are read-only, these work unit files will not be saved when you save your project. For more information on saving projects, refer to [Saving a Project](#).

To view the status of your project files:

1. Do one of the following:

From the menu bar, click **Project > File Manager**.

Press **Shift+F1**.

The File Manager dialog box opens.

2. Review the project files to verify whether:

They have been modified.

They are read-only.

3. Switch to the **Source Files** tab and review the source files to verify whether:

They have been modified.

They are read-only.

They are being used by one or more audio sources in your project.



Caution

The information displayed in the Usage column may not always be current. Before deleting any files that are marked “Unused”, you might want to close and re-open the project to make sure the information is completely up to date.

4. Click OK to close the Project File Status dialog box.

Related Topics

- [Saving a Project](#)
- [Resolving Project Inconsistencies](#)
- [Workgroup Tips & Best Practices](#)

Using Wwise with Your Source Control System

In your project development environment, you may already be using a source control system to effectively manage your assets and other types of project files. All Wwise project files, including the individual work units, are XML-based, which means you can use your source control system to easily manage these files as well.

The following files within your project can be managed by your source control system:

- **Wwise project file** - The WPROJ file.
- **Wwise work units** - The WWU files, including the Default work units.
- **Originals folder** - The folder that contains the original sound files that were imported into Wwise.
- **Generated SoundBanks** - The SoundBank files generated for each platform and language.



Caution

The .cache folder located in your project directory is a local working folder for Wwise. The contents of the .cache folder should not be added to your source control system because it may cause unexpected behaviors in Wwise.

Throughout the development of your game, you can view the status of your project file (.wproj), work unit files (.wwu), and audio files in the File Manager. If you are using the Perforce, Subversion, or another Workgroup plug-in, you will be able to perform source control functions directly in Wwise. For more information about using a Workgroup plug-in in Wwise, refer to [Managing Project Files Using a Workgroup Plug-in](#).

When you are working as part of a workgroup and are using a source control system to manage the files in your project, you should always be aware that others are working on the same project and that there may be merge conflicts that need to be resolved. This is why it is important to sync and merge your work often and to communicate frequently with your team members about the work you are doing. For a complete list of best practices, refer to [Workgroup Tips & Best Practices](#).

Resolving Project Inconsistencies

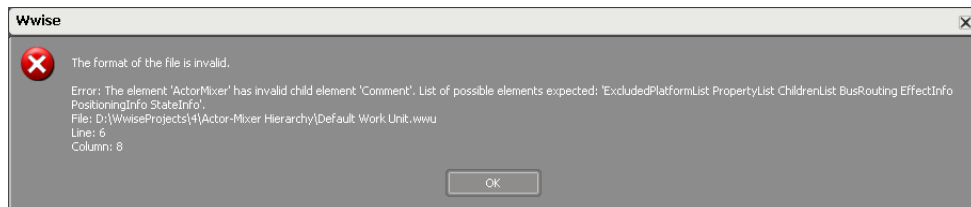
When several people are working on the same project, it is possible that changes to specific files may result in project file errors or inconsistencies. To reduce the impact of these issues, Wwise performs two types of project validation each time you open a project file:

- A validation for XML syntax and schema errors
- A validation for project inconsistencies

These validations can help you resolve errors and prevent further project problems.

XML Syntax and Schema Errors

When project files are checked-in or merged in your source control system, you may be required to update the XML code itself to resolve conflicts. If XML syntax or schema errors are created during the update, the project file will become invalid and Wwise will not be able to open the project. A message box is displayed that describes the error and specifies the exact file and location of the error.

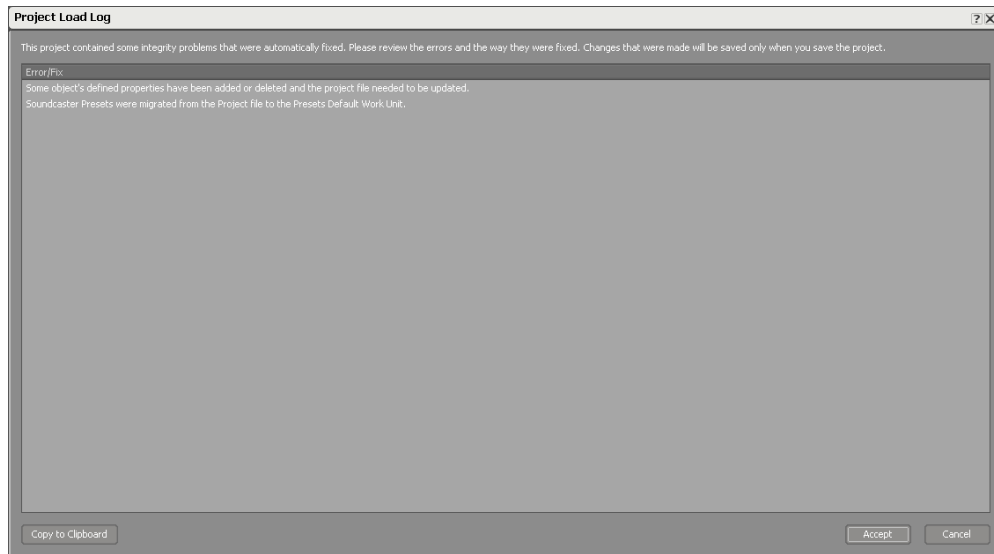


To resolve the problem, you will have to review the individual work unit files listed in the error.

Project Inconsistency Issues

If there are no XML syntax errors in your project files, Wwise goes on to verify if there are any project inconsistencies or issues. For example, a project inconsistency will occur if a state gets deleted in the States work unit, but is still being used by one of the objects in one of the sound or music structure work units.

If Wwise detects any project issues, it displays information about each issue along with a description of how it will be fixed, if necessary.



You have the choice to accept these fixes as a group or to reject them and either revert to older versions of the project, or try to fix the inconsistencies in the XML files themselves.

To resolve project inconsistencies:

1. Open a project in Wwise.

If project inconsistencies are found, the Project Load Log opens with a complete list of the project issues along with the fix recommended by Wwise, if necessary.

2. Do one of the following:

Click **Accept** to accept the recommended fixes.

Wwise fixes the inconsistencies and opens the project. If you accept the fixes, you should review each one individually to evaluate whether you need to fix things manually.

Click **Cancel** to reject the recommended fixes.

Wwise closes the project. You can either revert to an older version of the project or try to fix the inconsistencies in the XML files themselves.



Note

The fixes Wwise makes to your project are not saved until you save your project.

Managing Project Files Using a Workgroup Plug-in

Wwise's open architecture makes it easy for you to integrate your source control software by creating a Workgroup plug-in. The Workgroup plug-in creates a link between Wwise and your source control software so that you can manage your files and perform source control functions, such as check in and check out, directly in Wwise.

Because each source control system works differently, the specific functions available and the workflow you adopt will depend on the system with which you are working.

To help you get started, Wwise comes with two fully functional sample Workgroup plug-ins for the software configuration management systems Perforce® and Subversion. Perforce and Subversion both use the copy-modify-merge model as opposed to the lock-modify-unlock solution. In this model, each member of the workgroup reads the repository or depot and creates a client workspace which is a directory containing a personal working copy of the files and/or project on their workstation. This allows each member of your team to work in parallel, modifying their own private copies. When ready, each member can merge their private copies together into a new, final version. If there are conflicts, the version control system will assist you with the merging, but ultimately you are responsible for resolving any issues correctly.

If you are using another source control management system, you can create your own workgroup plug-in for Wwise. For information on creating and integrating your own workgroup plug-in, refer to the section [How to Create a Source Control Plug-in DLL](#) in the SDK documentation.

When using a source control plug-in, a number of different source control functions are available directly in Wwise. For example, when using the Perforce plug-in, you can perform the following source control operations from within Wwise:

- **Get latest version** to update your working copy by retrieving the latest version of the file from the depot.

- **Submit changes** to send pending files to the Perforce server for processing.
- **Check out** to make the working copy of the depot file in your client workspace.
- **Lock** to lock a file so that other clients can't submit their working copies of the file to the server.
- **Unlock** to unlock a file so that other clients can submit their working copies of the file to the server.
- **Mark for add** to add a file to the depot. When a file is marked for add, it is added to a changelist and then must be submitted to the depot.
- **Mark for delete** to delete a file from the depot. When a file is marked for delete, it is placed in a changelist and then must be submitted to the depot where it is deleted.
- **Move** to move a source file between folders within the Originals directory.
- **Rename** to rename a file in the depot.
- **Revert changes** to discard the changes you have made to a file in your client workspace.
- **Resolve** to reconcile the differences between two revisions of a file.
- **Diff** to compare the file in your client workspace with the file in the depot.
- **File History** to display a file's revision history.




Note

A list of similar source control functions exists for the Subversion plug-in.

Not only will you have access to these source control commands in Wwise, but special icon overlays are displayed in the Project Explorer to help you quickly identify the status of your work unit files. The following table displays the icon overlays used, for example, in the Perforce Workgroup plug-in.

Overlay Icon	Name	Description
	Normal (not checked out)	File is up-to-date and not checked out by anybody.
	Marked for add Moved	Files have been marked for add to the Perforce depot or has been moved with the move/add status.
	Checked out (Open for Edit)	Files have been checked out from the Perforce depot.
	Checked out by another user	Files have been checked out by another user in your workgroup.
	Concurrently checked out	Files have been checked out by you and by another user in your workgroup.
	Outdated (not latest revision of file)	A more recent revision exist on the server. You can get the latest version of the file.

Overlay Icon	Name	Description
		



Note

The icons “Checked out by another user”, “Concurrently Checked out” and “Outdated” are only available with the Perforce workgroup plug-in.

Supported Perforce/Subversion Versions

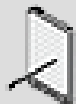
Before using either of the workgroup plug-ins, please review the following information to make sure that your version of Perforce or Subversion is compatible with the workgroup plug-in.

Perforce

The Perforce workgroup plug-in was built using the Perforce SDK version 2015.1, but should work with most versions of the Perforce servers. If you are using a different version of Perforce that is not compatible with the workgroup plug-in, you can use the sources in the Wwise SDK to rebuild the plug-in with the version you are using. For more information about the Perforce plug-in, refer to the [Wwise SDK documentation](#).

Subversion

The Subversion workgroup plug-in was built using Subversion version 1.8.5. If you are using a different version of Subversion that is not compatible with the workgroup plug-in, you can use the sources in the Wwise SDK to rebuild the plug-in with the version you are using. For more information about rebuilding the Subversion plug-in, refer to the [Wwise SDK documentation](#).



Note

Subversion's bin folder, which by default is located in C:\Program Files\Subversion\bin, must be in the system's PATH environment variable. Subversion installers normally add it to the PATH, but if Subversion is installed manually from the ZIP file mentioned above, for example, then you must add the bin folder to the PATH manually.

The following sections describe how to manage your assets using a workgroup plug-in. Although they primarily focus on the Perforce workgroup plug-in, a similar workflow exists when using Subversion.

- [Setting Up Your Project Files in Perforce](#)
- [Adding Files to Perforce](#)
- [Checking Out Files from the Perforce Depot for Editing](#)
- [Submitting Your Changes to the Perforce Depot](#)
- [Comparing Your Files with the Files in the Perforce Depot](#)
- [Moving Source Files within the Originals Folder Using Perforce](#)
- [Renaming Files When Using Perforce](#)
- [Deleting Files from Your Project When Using Perforce](#)
- [Saving Your Project When Using Perforce](#)

Setting Up Your Project Files in Perforce

Before you can start using the Perforce workgroup plug-in, you or your system administrator need to do the following:

- Install and set up the client version of the source control management software on the workstation of each person in your workgroup.
- Configure a client workspace on the workstation of each person in your workgroup.
- Get the latest version of the project files from the depot and save them in your client workspace.
- Select and configure your workgroup plug-in in Wwise.

For more information on installing and configuring Perforce, contact your system administrator, or consult the Perforce documentation.

You can select and configure the workgroup plug-in in the Project Settings dialog box. For more information on selecting and configuring the workgroup plug-in in Wwise, refer to [Configuring Source Control Plug-ins](#).

Related Topics

- [Adding Files to Perforce](#)
- [Checking Out Files from the Perforce Depot for Editing](#)
- [Submitting Your Changes to the Perforce Depot](#)
- [Comparing Your Files with the Files in the Perforce Depot](#)
- [Moving Source Files within the Originals Folder Using Perforce](#)
- [Renaming Files When Using Perforce](#)
- [Deleting Files from Your Project When Using Perforce](#)
- [Saving Your Project When Using Perforce](#)

Adding Files to Perforce

Before you start working in a project, you should add your project files and audio sources to the depot or repository. Adding files to Perforce consists of the following two steps:

- [To add files to the changelist.](#)
- [To submit your changes to the depot.](#)



Note

When you create new work units or import new audio files into your project, you will be prompted to mark these new files for addition to Perforce.

To add files to the changelist:

1. From the menu bar, click **Project > File Manager**.

The File Manager opens.

2. Switch to one of the following tabs:

Work Units to add work units to Perforce.

Sources to add audio files to Perforce.

3. Select the file or files that you want to add to Perforce.
4. Right-click the selected project file(s) and select **Mark for add** from the menu.

The Process Log dialog box opens showing that the selected file has been added to the changelist or “opened for add” and is ready to be submitted to the depot.

5. Click **OK** to close the Process Log.

The status of the files is set to “add”.

6. Click **Close** to close the File Manager.

In the Project Explorer, the project files now have the Marked for add overlay icon.

At this point, the files have only been added to the changelist. You need to submit the changelist to check the new files into the depot. For more information on submitting files to the depot, refer to [Submitting Your Changes to the Perforce Depot](#).

7. Save your project.

Related Topics

- [Checking Out Files from the Perforce Depot for Editing](#)
- [Submitting Your Changes to the Perforce Depot](#)

- [Comparing Your Files with the Files in the Perforce Depot](#)
- [Moving Source Files within the Originals Folder Using Perforce](#)
- [Renaming Files When Using Perforce](#)
- [Deleting Files from Your Project When Using Perforce](#)
- [Saving Your Project When Using Perforce](#)

Checking Out Files from the Perforce Depot for Editing

Before editing the files in your project, you should check them out first to make sure that you have the latest version of the files in your client workspace. When you check out files in Perforce, the files become writable. This means that you will be able to save the changes you made to a file.

By default, Perforce does not do an exclusive check out, which means that other members of your team may be working on the same files at the same time. Communication is key to avoid merge conflicts that can be time-consuming and difficult to resolve.

To check out files for editing:

1. From the menu bar, click **Project > File Manager**.

The File Manager opens.

2. Switch to one of the following tabs:

Work Units to add work units to Perforce.

Sources to add audio files to Perforce.

3. Select the file(s) that you want to open for edit.
4. Right-click one of the selected project files and select **Check out** from the menu.

The Process Log dialog box opens showing that the file has been checked out or “opened for edit”.

5. Click **OK** to close the Process Log.

The status of the files is set to “edit”.

6. Click **Close** to close the File Manager.

In the Project Explorer, the project file now has the Checked out overlay icon.

Related Topics

- [Adding Files to Perforce](#)
- [Submitting Your Changes to the Perforce Depot](#)

- [Comparing Your Files with the Files in the Perforce Depot](#)
- [Moving Source Files within the Originals Folder Using Perforce](#)
- [Renaming Files When Using Perforce](#)
- [Deleting Files from Your Project When Using Perforce](#)
- [Saving Your Project When Using Perforce](#)

Submitting Your Changes to the Perforce Depot

If you are finished working with a particular file or want others in your team to have access to your changes, you can submit the file back to the depot. It is very important that you give a detailed description of the changes you made so that you and others fully understand the extent of the changes made to the files. These comments can help when dealing with conflicts.

When you submit your files back to the depot, Perforce merges the changes you made with the file already in the depot. If any conflicts exist, they must be resolved before a file can be successfully submitted to the depot.

To submit your changes to the depot:

1. From the menu bar, click **Project > File Manager**.

The File Manager opens.

2. Click the **Submit Changes** button and then one of the following options:

Work Units - To send only the work units to the server for check-in.

Sources - To send only source files to the server for check-in.

All - To send all pending files (work units and source files) to the server for check-in.

The Submit Changes dialog box opens.

3. If you don't want specific files to be checked-in to the server, clear the selection for those files and then click **OK**.
4. Right-click one of the selected project files and select **Submit Changes** from the menu.

The Description dialog box opens.

5. Type a detailed description of the changes you made to the files.
6. If you want to continue editing the files again immediately after submitting your changes, select the **Check out file(s) after submit** option.
7. Click **OK**.

The Process Log dialog box opens showing that the files have been checked in or “submitted”.

8. Click **OK** to close the Process Log.

The status of the file is set to “normal”.

9. Click **Close** to close the File Manager.

In the Project Explorer, the project files will have the Checked in overlay icon, unless you selected the Check out file(s) after submit option.

To submit individual files to the depot:

1. From the menu bar, click **Project > File Manager**.

The File Manager opens.

2. Switch to one of the following tabs:

Work Units to add work units to Perforce.

Sources to add audio files to Perforce.

3. Select the file or files that you want to submit.
4. Right-click one of the selected project files and select **Submit Changes** from the menu.

The Description dialog box opens.

5. Type a detailed description of the changes you made to the files.
6. If you want to continue editing the files again immediately after submitting your changes, select the **Check out file(s) after submit** option.
7. Click **OK**.

The Process Log dialog box opens showing that the files have been checked in or “submitted”.

8. Click **OK** to close the Process Log.

The status of the file is set to “normal”.

9. Click **Close** to close the File Manager.

In the Project Explorer, the project files now have the Checked in overlay icon, unless you selected the Check out file(s) after submit option.



Tip

You can also submit changes to the perforce depot by right-clicking a work unit in the Project Explorer and selecting **Workgroup > Submit Changes**.

Related Topics

- [Adding Files to Perforce](#)
- [Checking Out Files from the Perforce Depot for Editing](#)
- [Comparing Your Files with the Files in the Perforce Depot](#)
- [Moving Source Files within the Originals Folder Using Perforce](#)
- [Renaming Files When Using Perforce](#)
- [Deleting Files from Your Project When Using Perforce](#)
- [Saving Your Project When Using Perforce](#)

Comparing Your Files with the Files in the Perforce Depot

You can compare or diff a file in your client workspace with the latest revision of the file in the depot to determine if there are differences between the two files. Before diffing a file, you should save your project so that you are using the latest version of a file for the diff.

To compare your file with a file in the depot:

1. From the menu bar, click **Project > File Manager**.

The File Manager opens.

2. Right-click the work unit file you want to compare, and select **Diff** from the menu.

The differences between the two files are displayed in the default Perforce diff viewer.

3. Review the differences between the two files.
4. When you are finished reviewing the differences, close Perforce to return to Wwise.

Related Topics

- [Adding Files to Perforce](#)
- [Checking Out Files from the Perforce Depot for Editing](#)
- [Submitting Your Changes to the Perforce Depot](#)
- [Moving Source Files within the Originals Folder Using Perforce](#)
- [Renaming Files When Using Perforce](#)
- [Deleting Files from Your Project When Using Perforce](#)
- [Saving Your Project When Using Perforce](#)

Moving Source Files within the Originals Folder Using Perforce

You can move the source files within your project's Originals folder, but before doing so you should make sure that no other team members are working on the

files. If you do move a file while others have the file checked out, they will run into problems when they try to submit their changes to the depot.

When Perforce moves a file, it opens the current file for delete and creates an exact copy of the file in the new location. These files are not automatically updated in the depot, so after the move is complete, you must submit the changelist to the depot. Before you can move a file, it must be checked into the depot.



Note

You must save your project before moving the source files in the Originals folder.

To move files between folders within the Originals directory:

1. Save your current project.
2. Do one of the following:

From the menu bar, click **Project > File Manager**.

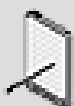
Press **Shift+F1**.

The File Manager opens.

3. Switch to the **Source Files** tab.
4. Select the file(s) you want to move.
5. Right-click the selection and select **Move** from the menu.

The Browse For Folder dialog box opens.

6. Navigate through the folder structure and select the folder where you want to move the file.



Note

If you want to create a new folder, select the parent folder, click the Make New Folder button, and then specify a name for the folder.

7. Click **OK** to move the file to the new folder.

The Process Log dialog box opens showing that a new copy of the file has been added to the specified folder using the branch command and the original file has been 'opened for delete'.

8. Click **OK** to close the Process Log dialog box.

9. Select both entries in the list; the branch and deletion.
10. Right-click the selection and select **Submit Changes** from the menu.

The Description dialog box opens.

11. Type in a useful description and then click **OK**.

The Process Log opens showing that both pending operations were submitted to the depot.

12. Click **OK** to close the Process Log.

The file now resides in the specified folder.

Related Topics

- [Adding Files to Perforce](#)
- [Checking Out Files from the Perforce Depot for Editing](#)
- [Submitting Your Changes to the Perforce Depot](#)
- [Comparing Your Files with the Files in the Perforce Depot](#)
- [Renaming Files When Using Perforce](#)
- [Deleting Files from Your Project When Using Perforce](#)
- [Saving Your Project When Using Perforce](#)

Renaming Files When Using Perforce

You can rename a file in your project, but before doing so you should make sure that no other team members are working on the file. If you do rename a file while others have the file checked out, they will run into problems when they try to submit their changes to the depot.

When Perforce renames a file, it opens the current file for delete and creates an exact copy of the file with the new name. These files are not automatically updated in the depot, so after the rename is complete, you must submit the changelist to the depot. Before you can rename a file, it must be checked into the depot.



Caution

Do not rename the default work units as they are critical to the project.

To rename a file in your project:

1. From the menu bar, click **Project > File Manager**.

The File Manager opens.

2. Right-click the work unit file you want to rename, and select **Rename** from the menu.

The Rename File dialog box opens.

3. In the Name field, type the new name for your file.
4. Click **OK**.

The Process Log dialog box opens showing that a copy of the file was created with the new name using the branch command and the old file was opened for delete.

5. Click **OK** to close the Process Log.

The status of the old file is set to “delete”, and a new file has been created with the status set to “branch”.

6. Select both the old and new files.
7. Right-click and select **Submit Changes** from the menu.

The Description dialog box opens.

8. Type in a useful description of the change and click **OK**.

The Process Log opens showing the deletion of the original file and the addition of the newly renamed file.

9. Click **OK** to close the Process Log.
10. Click **Close** to close the File Manager.

The Project dialog box opens prompting you to reload the latest version of the project because changes have been made externally to the project.

11. Click **Yes**.

The latest version of the project is loaded and the file has been renamed.

Related Topics

- [Adding Files to Perforce](#)
- [Checking Out Files from the Perforce Depot for Editing](#)
- [Submitting Your Changes to the Perforce Depot](#)
- [Comparing Your Files with the Files in the Perforce Depot](#)
- [Moving Source Files within the Originals Folder Using Perforce](#)
- [Deleting Files from Your Project When Using Perforce](#)
- [Saving Your Project When Using Perforce](#)

Deleting Files from Your Project When Using Perforce

When you no longer require certain files in your project, you can delete them. When you delete a file in Perforce, you must first open it for deletion and then

submit the changelist to the depot. You should always save your project before deleting files to prevent losing the changes you made since the last save.



Caution

Do not delete the default work units as they are critical to the project.

To delete a file from your project:

1. From the menu bar, click **Project > File Manager**.

The File Manager opens.

2. Select the file(s) that you want to delete.
3. Right-click one of the selected project files, and select **Mark for Delete** from the menu.

A message is displayed prompting you to confirm the file deletion.

4. Click **Yes**.

The Process Log dialog box opens showing that the file(s) has been opened for delete.

5. Click **OK** to close the Process Log.

The status of the file is set to “delete”.

6. Click **Close** to close the File Manager.

The Project dialog box opens prompting you to reload the latest version of the project since changes have been made externally to the project.

7. Click **Yes**.

The latest version of the project is loaded, and the file has been deleted.

At this point, the file has only been added to the changelist and is marked for delete. You need to submit the changelist to delete the file in the depot.

Related Topics

- [Adding Files to Perforce](#)
- [Checking Out Files from the Perforce Depot for Editing](#)
- [Submitting Your Changes to the Perforce Depot](#)
- [Comparing Your Files with the Files in the Perforce Depot](#)
- [Moving Source Files within the Originals Folder Using Perforce](#)
- [Renaming Files When Using Perforce](#)

- [Saving Your Project When Using Perforce](#)

Saving Your Project When Using Perforce

As long as you have checked out all the project files that you plan on changing, you won't have any problems saving your project. If, however, you make changes to files that are not checked out, Wwise will prompt you to check these files out before it can save the entire project.

To save a project:

1. Do one of the following:

From the menu bar, click **Project > Save**.

Press **Ctrl+S**.

Wwise attempts to save all files in the project. If you made changes to files that were not checked out, the Pending Source Control Operations dialog opens.

2. Select the **Check out modified files** option to check out the files that have changed so that Wwise can save them.
3. Click **OK**.

Wwise checks out the necessary files and saves the project.



Note

Click **Cancel** to stop the save operation altogether.

Related Topics

- [Adding Files to Perforce](#)
- [Checking Out Files from the Perforce Depot for Editing](#)
- [Submitting Your Changes to the Perforce Depot](#)
- [Comparing Your Files with the Files in the Perforce Depot](#)
- [Moving Source Files within the Originals Folder Using Perforce](#)
- [Renaming Files When Using Perforce](#)
- [Deleting Files from Your Project When Using Perforce](#)

Workgroup Tips & Best Practices

Before using a Source Control System with Wwise, you may want to review the following sections, which provide you with a series of tips and best practices

that can help you better manage your team and project files throughout the audio development process.

Planning Your Project

- Divide your project into smaller work units - if your projects are large and you keep all your project data in the Default work units, you will not only slow down Wwise's response time, but the members of your team will need to merge every time someone makes a change, which can greatly increase the chance of merge problems and conflicts. By dividing up your project into smaller chunks using work units, people can work more efficiently, can access information more quickly, and can work on different areas, which can reduce the possibility of merge problems.
- Assign responsibility for global default work units - some project elements, such as Master-Mixer hierarchy and presets, can't be sub-divided into additional work units. It may be a good idea for one person to manage or at least be aware of the changes that are being made to these project elements.

Basic Workflow

- Manage global project elements effectively - when you rename or delete a global project element, such as a state or game parameter, be aware that you might be modifying many other objects within your project, including all the sound objects and containers that are using these elements. When you save and check-in these changes, you might be affecting a number of different project files that other people are working on. To limit the impact of this type of change, you should define the global project elements as early as possible and then try to avoid making changes to them after that. If changes are required after the initial setup, you should do the following:
 - Warn your team members that a global element has changed.
 - Ask each team member to check-in their changes.
 - Check-in your files.
 - Ask everyone on your team to update their project files.
 - By following this process, the members of your team will only have to update to get the new files, without having to merge.
- Check project file status - before you begin working on a work unit, use the File Manager to verify which files are read-only. If you change a work unit that is read-only, you will not be able to save that particular file within your project.
- Backup your local files regularly - although the files in the central repository may be part of a scheduled backup plan, the copies on your local machine are not. To prevent data loss, it is a good idea to backup your project files on a regular basis, especially if you have made significant changes to the files.
- Generate integrity report before checking in - before checking in a particular work unit, it is a good idea to generate the integrity report to make sure there

aren't any project errors. If errors exist, you can quickly resolve them and then check the work unit in.

Syncing Your Files

- Sync before new work sessions - you should sync your project files with the server before starting a new work session so that you have the latest modifications.
- Close Wwise before syncing - before syncing your project files with the server, you should close Wwise to prevent the possibility of losing information. If Wwise is open, a copy will remain in memory. When you sync, the files on your disk will be modified, but the older version of your project will remain in memory. If you save the currently opened project, you can overwrite the changes of others that were saved to your disk.

This only applies if you are not using a workgroup plug-in. When you sync using a workgroup plug-in, you are automatically prompted to reload the latest version of the project.

Checking in Your Files

- Submit often - if you have large changes that affect other members of your team, you should submit your project files often to the server so that others can benefit from the changes you are making. If you wait too long, you also increase your chances of conflicts. By submitting smaller, targeted change, it will be easier to revert to an older version of the project if necessary and easier to resolve conflicts if any arise.
- Add useful comments - when you commit or check-in your files, make sure to fully describe the changes that you made.

Source Control System

- Understand your source control system - before using a source control system to manage your Wwise project files, you should have a good understanding of how it works. Knowing the intricacies of your source control system can help you avoid problems and create an effective and efficient production pipeline.

Project Inconsistencies

- Become familiar with Wwise project XML structure - spend some time familiarizing yourself with the XML structure of the Wwise project files before merging your files. In some circumstances, you may have to update the XML code, so if you don't understand it correctly, you may break your project. If you do modify the XML code, make sure to open the project in Wwise before checking the files back in to your source control system. This will ensure that the changes you made to the XML code are valid and what you wanted.

Communication

- Encourage frequent and open communication - in any workgroup environment, it is key that you communicate openly and frequently with the other members of your team. Frequent and open communication can result in fewer conflicts, less time merging files, and a more efficient production pipeline. For example, before making changes to a work unit that effects other members of your team, you should ask them to check in their changes and wait for your word before syncing and resuming their work.

File Usage

- Before deleting any files that are marked “Unused” in the Usage column of the File Manager, it is a good practice to close and re-open the project to make sure the information is completely up to date.

SoundBanks

- To avoid having to edit your SoundBanks each time changes are made to your project, you can re-create the way your SoundBanks are set up using work units and/or folders. Since Wwise maintains an active link between the elements in a SoundBank and those in your project, once these work units have been added to the SoundBank you will never have to edit the SoundBanks again because they will get updated automatically.

Chapter 6. Managing Media Files in Your Project

Overview	143
The Importing Process	143
Importing Media Files	151
Replacing Media Files	167
Managing File Import Issues	170
Re-Organizing Media Files within the Originals Folder	172
Clearing Your Cache	173
Editing Audio Files in an External Editor	174
Creating Audio and Motion Sources Using Plug-ins	176
Media File Management Tips and Best Practices	177

Overview

In a typical game you can have thousands of assets and it is important to know how to manage them efficiently in your project. At a very basic level, Wwise distinguishes between the following two types of assets:

- Original assets imported into Wwise.
- Versions of these assets created for the various game platforms.

Wwise stores these two types of assets in different locations within your project folder so they can be managed independently. The original assets are stored in the “Originals” project folder. Since these assets are usually shared by several people on your team, this folder can be located anywhere on your network and can easily be managed by your source control system.

The asset versions created for the various game platforms are stored locally in each user's project cache folder. This allows each user to manage their own platform versions and to experiment with different conversion settings.

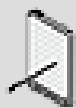
Both types of files can be played back in Wwise. The original pre-converted files will be played back whenever the Original control is activated in the Transport Control or Soundcaster. When the Original control is not active, Wwise will attempt to play the converted file, if one exists. It is important to note, however, that there are some restrictions to playing converted files in Wwise. When you convert an audio file for a particular platform, it is converted to meet the specific hardware requirements of that platform. As a result, you may not be able to play back these converted files in Wwise when a platform, other than Windows, is selected.

To test your converted files in Wwise, you can do the following:

- PCM - Select the Windows platform.
- ADPCM - Select the Windows platform.
- Vorbis - Select the Windows platform.
- XMA - Select the Xbox 360™ platform.

The Importing Process

Before importing your files into your Wwise project, you should first look at what kinds of files are supported by Wwise, and where you want these media files to be stored.



Note

The length of the filenames that can be imported into Wwise is 256 characters. This number includes the complete path of the

filename. For more information about filename limits, refer to [Understanding File Length Limitations in Wwise](#).

What Media Files are Supported?

When looking at what type of media files are supported in Wwise, you need to look at the following:

- [PCM Audio File Format](#)
- [Channel Configuration](#)
- [Sample Rate](#)
- [Bit Depth](#)



Note

Trying to import files that are not supported by Wwise will result in non-recoverable errors that are displayed in the Import Conflict Manager. For information on how you can deal with non-recoverable errors, refer to [Managing Non-Recoverable Errors](#).

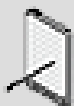
PCM Audio File Format

Only files in the Waveform audio format (.wav) can be imported into Wwise.

Channel Configuration

Wwise can import media files with up to 8 different audio channels, ranging from 0.1 to 7.1.

The following table lists each of the source channel configurations supported by Wwise and the default configuration used by Wwise.



Note

Wwise follows the Microsoft standard for channel ordering. The ordering in the file is different from the display ordering in Wwise. The table below shows the ordering in the file.

Channel Short Names	Channel Names and angles
L	Left (22-30 degrees)
R	Right (22-30 degrees)
C	Center (0 degree)
SL	Surround Left (90-110 degrees)

Channel Short Names	Channel Names and angles
SR	Surround Right (90-110 degrees)
BL	Back Left (135-150 degrees)
BR	Back Right (135-150 degrees)
LFE	Low-Frequency Effects

Channels	Channel Configuration Used by Wwise
0.1	LFE
1.0	C
1.1	C + LFE
2.0	L + R
2.1	L + R + LFE
3.0	L + R + C
3.1	L + R + C + LFE
4.0	L + R + SL + SR
4.1	L + R + LFE + SL + SR
5.0	L + R + C + SL + SR
5.1	L + R + C + LFE + SL + SR
6.0	L + R + BL + BR + SL + SR
6.1	L + R + LFE + BL + BR + SL + SR
7.0	L + R + C + BL + BR + SL + SR
7.1	L + R + C + LFE + BL + BR + SL + SR

If you have a specific multichannel configuration that you want Wwise to preserve on import, you must define it in the WAVEFORMATEXTENSIBLE format chunk in the WAV header. If the channels defined in the channel mask of the WAVEFORMATEXTENSIBLE are not supported by Wwise, they will be changed automatically to supported values. For example, if a configuration defines a Left channel but no right channel, the Left channel (0x01) is changed to the Center channel (0x04), and if a configuration defines Side (Surround) channels, but no Rear (Back) channels, the Side channels (0x100 and 0x200) are changed to Rear channels (0x10 and 0x20).

You can combine individual mono and stereo audio files into a single multi-channel file using the Wwise Multi-Channel Creator application. This standalone utility is installed as part of the Wwise package and can be found in the same directory as the Wwise application. The Multi-Channel Creator allows you to create files with any multi-channel configuration so that they can be imported into your Wwise projects. You can also use this tool to flag specific mono files as the 0.1 component so they can be imported into Wwise as the independent LFE channel.

Sample Rate

Wwise can import media files with sample rates up to and including 96 kHz.

Bit Depth








Wwise can import media files with the following bit depths:


- 16-bit
- 24-bit


Creating Wwise Objects on Import

After you have verified that the files that you want to import are supported by Wwise, you can determine what kind of object that you want to create with the files.

Wwise distinguishes between voice and SFX objects. SFX objects include sound objects and music objects called music segments. Since voice objects may be translated into different languages, they are handled differently in Wwise. To help you differentiate between voice and SFX objects, Wwise uses different icons to represent each object type.

Icon	Represents
	<p>SFX - Created by default when you import media files into the Actor-Mixer hierarchy.</p> <div data-bbox="511 987 1412 1155"><p>Tip</p><p>Hold the Ctrl key while dragging a WAV file to the Project Explorer to automatically create a Sound SFX object.</p></div>
	<p>Voice - Created for the dialogue that is likely to be translated for various language versions of the game.</p> <div data-bbox="511 1255 1412 1423"><p>Tip</p><p>Hold Ctrl and Shift keys while dragging a WAV file to the Project Explorer to automatically create a Sound Voice object for the Reference Language.</p></div>
	<p>Music segment - Created by default when you import media files into the Interactive Music hierarchy.</p> <div data-bbox="511 1528 1412 1696"><p>Note</p><p>When dragging a WAV file on an existing Music Segment, music tracks will be created for every WAV file dragged.</p></div> <div data-bbox="511 1724 1412 1892"><p>Tip</p><p>Hold the Ctrl key while dragging a WAV file to the Project Explorer in the Interactive Music Hierarchy to automatically create a Music Segment object.</p></div>

Icon	Represents
	Random Container - Created by default when dragging a folder into the Actor-Mixer hierarchy.



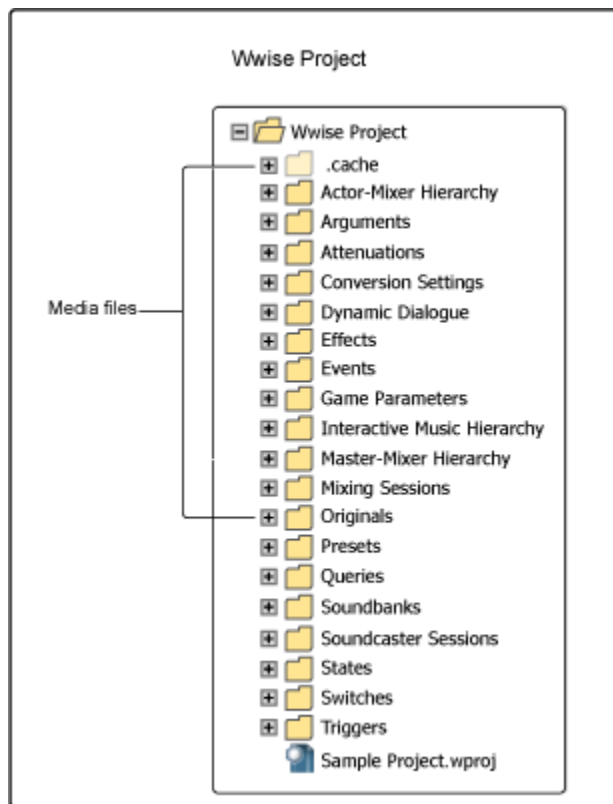
Tip

Hold the Ctrl key while dragging a folder to the Project Explorer in the Actor-Mixer Hierarchy to automatically create a Random Container with Sound SFX objects for every WAV file found in the folder.

The Media File Structure

It is helpful to understand how media files are stored in a Wwise project, particularly if you are working as part of a team. Media files are stored in two separate folders:

- Originals folder
- .cache folder



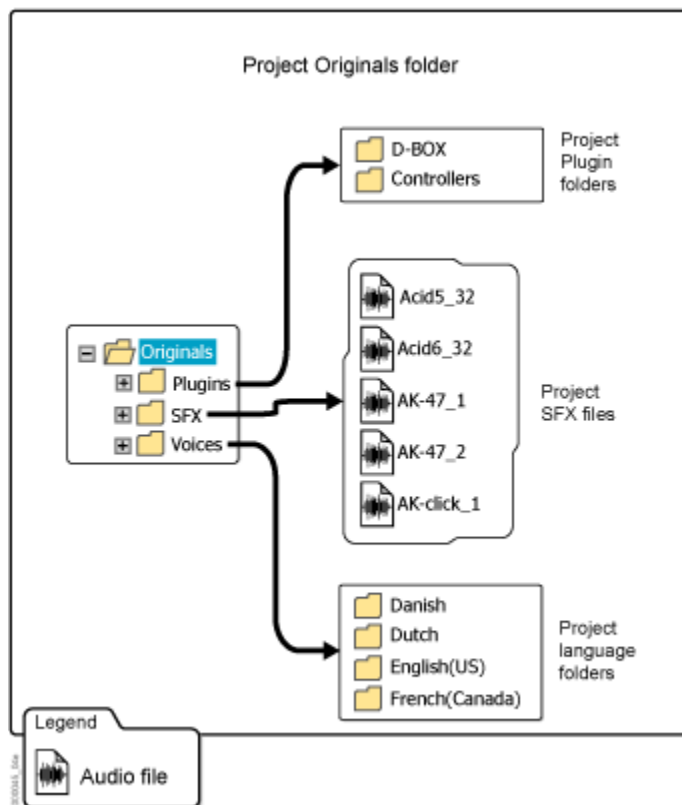
Originals Folder

When you import media files into Wwise, they are copied to the Originals project folder and remain unconverted and unchanged. Generally this project folder is stored under source control in a location that is accessible to all people working on the project. The location of the Originals folder is specified when

the project is created, but you can change the location in the Project Settings dialog box. These “original” files are played back in Wwise when you select Original in the Transport Control or Soundcaster.

The Originals folder is sub-divided into the following folders:

- Plugins
- SFX
- Voices



All media files imported into Wwise are categorized by type, so that Wwise can manage them effectively.

As your project grows in size, you may want these files within these folders to be organized into sub-folders. If this is the case, you need to be aware of the following restrictions before importing your files:

- Media files organized into sub-folders need to be imported into Wwise from within the sub-folder. This allows Wwise to manage the location of each media file by saving the media file using its full path and name.
- Each file path must be unique. This means that all media file names within a particular folder must be unique. This does not, however, prevent two media files from having the same name, they just have to reside within a different folder.

If you have already imported your media files into the project, you can also move your source media files between folders. For more information on moving your source media files, refer to [Re-Organizing Media Files within the Originals Folder](#).

.Cache Folder

By default, the cache folder is stored within your Wwise project and contains the converted media files for each of the platforms that you are developing for your game. These converted files are played back in Wwise when the Original button is not selected in the Transport Control or Soundcaster.



Note

You can modify the location of the cache folder in the Project Settings dialog box. For more information on modifying the location of your project cache folder, refer to [Defining Cache Folder Settings](#).

It is important to note, however, that there are some restrictions to playing converted files in Wwise. When you convert an audio file for a particular platform, it is converted to meet the specific hardware requirements of that platform. As a result, you may not be able to play back these converted files in Wwise when a platform, other than Windows, is selected.

To test your converted files in Wwise, you can do the following:

- PCM - Select the Windows platform.
- ADPCM - Select the Windows platform.
- Vorbis - Select the Windows platform.
- XMA - Select the Xbox 360 platform.



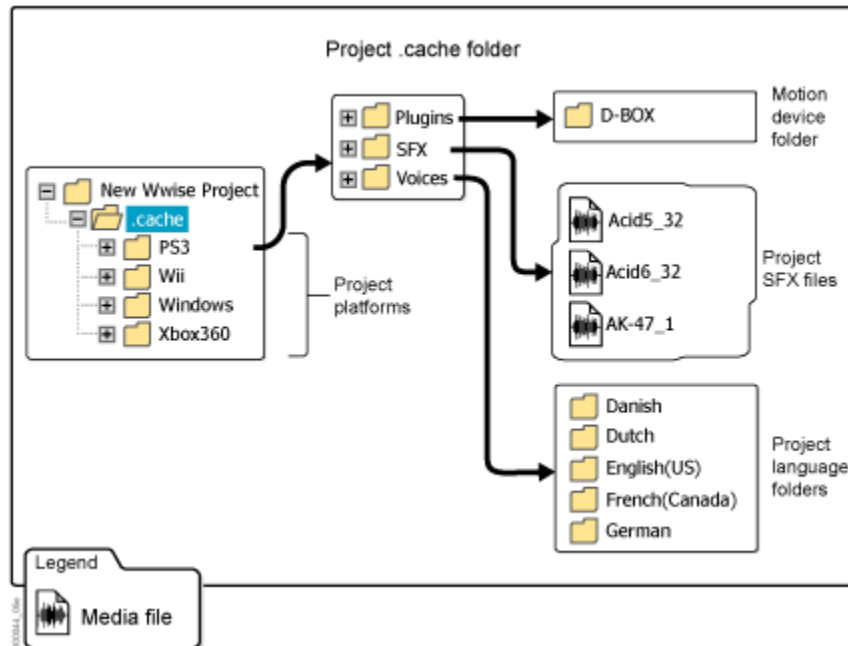
Caution

Do not include this folder in your source control system.

Each platform folder is further sub-divided into the following folders:

- Plugins
- SFX
- Voices

Wwise manages the contents of these folders for you, so you will never have to make any changes directly in these folders.



Note

The cache folder also contains a file called CacheVersion. This file includes information about the version of your cache. When you open a project, Wwise looks for the file and verifies to make sure that the version number of the cache matches the version number of Wwise. If no file exists or if the cache version is different than Wwise, the cache will be deleted and you will have to re-convert the platform version files before they can be played back again in Wwise.

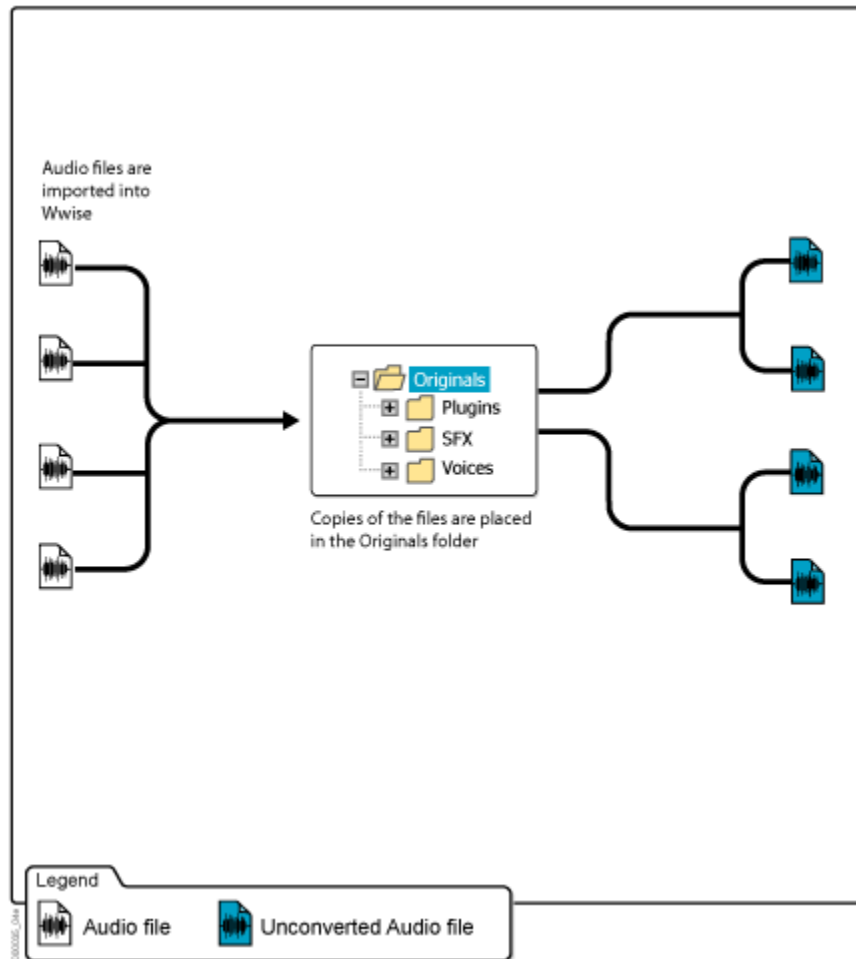
What Happens During the Importing Process?

Wwise carries out several activities during the import process. The conversion of media files for platforms is carried out separately. For more information on converting media files for each platform, refer to [Converting Audio Files](#).

The import process includes the following operations:

- The original media files are validated by Wwise before being copied into the project's Originals folder.
- Audio sources are created for the media files.
- Sound or music objects that contain the sources are created in Wwise and are displayed in their respective hierarchies in the Project Explorer's Audio tab.

The following illustration demonstrates what happens when you import media files into your Wwise project.



Importing Media Files

You will need to import your SFX or voice media files into Wwise at different times and for different reasons, depending on how you intend to use these media files. Usually you would import files in the following situations:

- To bring media files into your project at the beginning of the project, or as the files become available.
- To replace media files that you have previously imported. For example, you might want to replace placeholders or temporary files that you used at the beginning of the project.
- To bring language media files into the project for localization. For more information on localization, refer to [Localizing Your Project](#).

In Wwise, you can import the media files into your project using the Audio File Importer, or you can use the Quick Import shortcuts to carry out your audio file imports.

Quick Import

The following table lists the shortcuts you can use to import media files into Wwise.

To	Use this shortcut
Import SFX media files.	Drag the files into Wwise.
Import voice media files.	Shift+drag the files into Wwise.
Import SFX media files without opening the Audio File Importer.	Ctrl+drag the files into Wwise.
Import sound voice media files without opening the Audio File Importer	Ctrl+Shift+drag the files into Wwise. If you are importing languages using this shortcut, make sure that the reference language is selected in the Language Selector.
Import media files into the Contents Editor as new sources.	Drag the files into the Contents Editor.



Tip

If you use a spreadsheet program to manage the thousands of voice assets in your project, you can use the Voice Asset Importer to quickly create the corresponding sound voice objects in Wwise. For more information on importing voice assets into Wwise, refer to [Importing Voice Assets from a Text File](#).

Understanding File Length Limitations in Wwise

You can import files into Wwise with filename lengths of up to 256 characters. This file length limitation is actually a limitation of Windows and relates to the entire path of the filename. For example, a filename at the root of a directory can have up to a maximum of 252 characters. If the file is within a folder, the length of the folder name is taken into account as well. This limitation can cause problems when importing or converting media files in Wwise.

If you try to import media files that are near or at the maximum limits, you may experience some import problems. If the filename length is too long, either of the following errors will be displayed:

- Not a valid WAV file.
- Error copying the file to the originals folder.

In some cases, Wwise will allow you to import files that are near or at the maximum limit, but you may then encounter problems converting the files. In this case, the following error message will be displayed:

- Can't open source or output file.

To work around these issues, you will need to either reduce the length of the filename or reduce the overall length of the filename path.

Importing Media Files for SFX

In the Audio File Importer, you can specify that you want to import media files as SFX objects. When you import these files, SFX objects will be created in the selected location on the Audio tab of the Project Explorer. Depending on the hierarchy into which you are importing, these objects can be sound or music objects.

To import SFX media files into your project:

1. Do one of the following:

From the Wwise menu bar, click **Project > Import Audio Files**.

Right-click an object in the Actor-Mixer or Interactive Music hierarchies and select **Import Audio Files** from the menu.

The Audio File Importer opens.

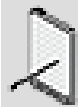
2. Click **Add Files** or **Add Folders**.

The Open File dialog box opens.

3. Browse to the location of the media files that you want to import.
4. Select the files or folders, and click **Open** or **Select Folder**.

The selected files or folders are loaded into the Audio File Importer.

5. To import media files as sound objects, select the **Import as Sound SFX** option.
6. To change the location where the objects will be created in Wwise, click [...] and select the new location in the hierarchy.



Note

To import media files as music objects, browse to the location in the Interactive Music hierarchy where you want to create the music objects. The **Import as SFX** and **Voice** options become unavailable.

7. Click **Import**.

The Importing dialog box opens where you can view the progress of the file import process.



Note

If there are errors in the media files, or conflicts, the Import Conflict Manager opens. For more information on how to deal with these conflicts, refer to [Managing File Import Issues](#).



Note

If you are using source control, you will be prompted to add these files to the source control system.

Related Topics

- [Importing Media Files for Voice-Overs](#)
- [Understanding File Length Limitations in Wwise](#)
- [Quick Import](#)
- [Replacing Media Files](#)
- [Managing File Import Issues](#)
- [Re-Organizing Media Files within the Originals Folder](#)
- [Clearing Your Cache](#)
- [Editing Audio Files in an External Editor](#)

Importing a Folder containing Media Files

It is possible to directly import one or multiple folder structures containing media files into your project. When you import a folder structure:

- Wwise can optionally create container objects for each folder in the imported structure.
- Wwise will copy the folder structure when copying the media files to the Original folder.

To import a folder structure into your project:

1. Do one of the following:

From the Wwise menu bar, click **Project > Import Audio Files**.

Right-click an object in the Actor-Mixer or Interactive Music hierarchies and select **Import Audio Files** from the menu.

The Audio File Importer opens.

2. Click **Add Folders**.

The Select Folder dialog box opens.

3. Browse to the location of the media files that you want to import.
4. Select the folders, and click **Select Folder**.

The selected folders are loaded into the Audio File Importer.

5. In the *Object Type/Action* column, select the object type you want to create for each folder you are importing.



Tip

You can select multiple folder entries in the Audio File Importer and then select the Object Type/Action to quickly change those entries simultaneously.

6. Click **Import**.

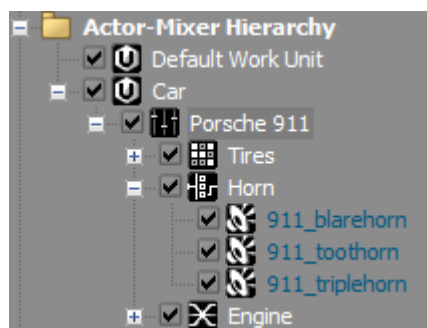
The Importing dialog box opens where you can view the progress of the file import process.

Using a Template to Import Media Files

When importing media files from a folder, you can select an object in your project that will be used by the Audio File Importer as a template to create new Wwise objects for the folders and media files you are importing. This can be useful when creating several similar structures where only the media files differ from structure to structure.

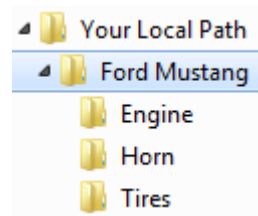
A typical example is when you want to create multiple car structures; they are all similar but have different media files and are associated with a specific car brand or model. Here is the workflow:

1. Create the first car structure in your project that will become your template to import the next structures. This structure can be a combination of any containers (actor-mixer, blend, switch, sequence, or random container) and sound objects.



If you are satisfied with it, then this car structure can be the template for your next one.

2. On your computer, create a folder structure that parallels the car structure you just created in Wwise to hold your media files (WAV and MID).



Now you have a structure that you will easily be able to import.

3. Click *Import Audio Files...* from the **Project** or the shortcut menu. Default shortcut: Shift+I.

The *Audio File Importer* opens.

4. Click **Add Folders....**

An Explorer dialog opens.

5. Navigate to and click the folder you want to import (the parent folder of the car structure you just created).

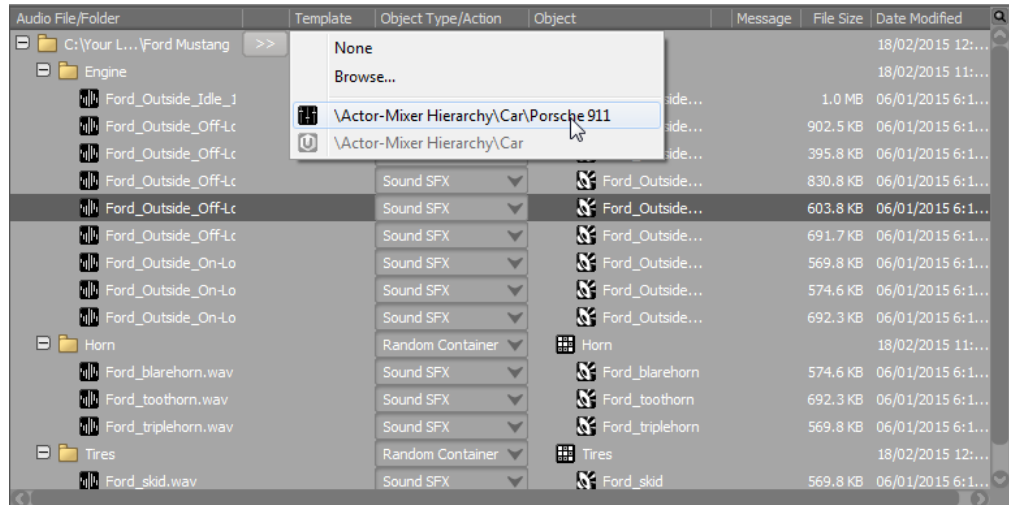
The selected folder's name appears in the **Folder** field.

6. Click **Select Folder**.

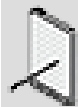
Focus returns to the **Audio File Importer** where the local folder hierarchy now appears in the **Audio File/Folder** panel, along with any media files (WAV or MID) that were found in the structure.

7. To the left of the *Template* column, click the selector [**>>**] and browse for the structure you created in the first step. This will set the template. The selector [**>>**] also lists the objects most recently used as templates.

Managing Media Files in Your Project



At this point, Wwise will try to automatically match the media files and folders you are importing with the objects in the template. The matching folders will be designated as the same container type as their template counterpart, while ones without a match will by default be Random Containers.



Note

You can change how strictly Wwise matches these imports by selecting from the **Template match mode** list in the lower left of the **Audio File Importer**. *Match all* mode uses special algorithms to find the best matches between the imported files and the template objects, while *Perfect match only* mode will only match imported files and folders with template objects when their names are exact matches. If Wwise finds no match between an imported item and the template, "No matching template found" appears in the **Message** column.



Tip

The template matching algorithm is based on name comparison. Therefore, best practice would be to have names in the folder hierarchy similar to the names in the template hierarchy.



Note

If the template did not match the structure being imported, you can completely remove the template by clicking the selector [>>] and selecting *None*. You can also disassociate a template by selecting a new object type in the *Object Type/Action* column.

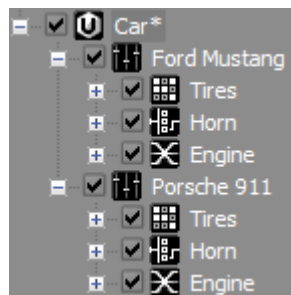
8. Click *Import*.



Note

If there is a conflict with existing files, the *Import Conflict Manager* opens to allow you to [resolve the conflict](#) before finalizing the import.

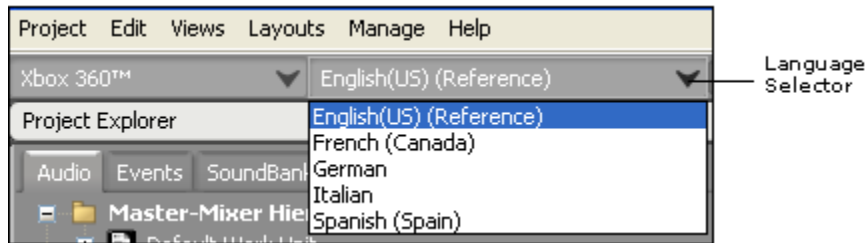
The **Audio File Importer** closes and the new structure appears in the **Project Explorer** hierarchy.



Importing Media Files for Voice-Overs

When you import media files for dialogue or voice-overs, you need to specify that these will be voice objects so that Wwise can later recognize these object in the localization process.

In some cases you will be importing voice files for localizing the project. When you import these files, they are saved in the project language folders and new sources are created in Wwise for these languages. Before you can do this, you need to define your project languages in the Language Manager. After you have set up your project languages, you can import the media files for your language sources. For more information on working with project languages, refer to [Localizing Your Project](#).



To import media files as voice objects into your project:

1. Do one of the following:

From the Wwise menu bar, click **Project > Import Audio Files**.

Right-click a Sound Voice or a Work Unit in the Actor-Mixer Hierarchy and select **Import Audio Files** from the menu.

The Audio File Importer opens.

2. Click **Add Files** or **Add Folders**.

The Open File dialog box opens.

3. Browse to the location of the media files that you want to import.
4. Select the files, and click **Open**.

The selected files are loaded into the Audio File Importer.

5. Select the **Import as Sound Voice** option.
6. To change the location where the objects will be created in Wwise, click [...] and select a new location in the hierarchy.
7. Click **Import**.

The Importing dialog box opens where you can view the progress of the import process.



Note

If there are errors in the media files or conflicts, the Import Conflict Manager opens. For more information on how to deal with these conflicts, refer to [Managing File Import Issues](#).

Related Topics

- [Managing the Languages for Your Project](#)
- [Importing Media Files for SFX](#)
- [Understanding File Length Limitations in Wwise](#)
- [Replacing Sound Voice Media Files](#)

- [Replacing SFX Files](#)
- [Managing File Import Issues](#)
- [Quick Import](#)
- [Re-Organizing Media Files within the Originals Folder](#)
- [Clearing Your Cache](#)
- [Editing Audio Files in an External Editor](#)

Importing Media Files from Tab Delimited Text File

It is possible to import a large amount of media files from a tab-delimited (also known as tab-separated) text file. Tab-delimited text files (.txt or .tsv extensions) can be generated by Microsoft Excel or other spreadsheet applications.

The tab-delimited file can define the following elements:



- WAV or MIDI file to import.
- Object structure to contain the files.
- Any property values or references, such as the Voice Volume or Output Bus.
- Events to create.

The first row of the tab delimited text file defines the content of the columns for the subsequent rows. The order of the columns has no impact on the import process.


Possible column headers are:

Header	Content
Audio File	<p>Defines the path to the WAV or MIDI file to import. The file will be copied to the Originals folder if it's not already present in it.</p> <p>The path can be absolute or relative to the tab- delimited file's directory.</p> <p>Example: C:\MyWaves\MyFolder\MySound.wav</p> <p>This column can be left blank if creating a container.</p> <p>If the column Object Type is not defined, the Audio File column defines the name of the imported object.</p>
Object Path	<p>Defines the path and name of the object to be created. The path used backslashes. The path can either be absolute or relative.</p> <ul style="list-style-type: none">• Absolute: full path, starting by a backslash and the object category. <p>Example: \Actor-Mixer Hierarchy\MyWorkUnit\MyVirtualFolder\MySound</p> <ul style="list-style-type: none">• Relative: path relative to the import destination, specified in the Audio File Importer dialog. <p>Example: MyVirtualFolder\MySound</p> <p>If this column is left blank, the name of the object will be taken from the imported audio file, and will be imported to the destination specified in the Audio File Importer dialog.</p>


Managing Media Files in Your Project

Header	Content
Object Type	<p>Specifies the type of the object.</p> <p>Common object types:</p> <ul style="list-style-type: none"> • Virtual Folder • Actor-Mixer • Switch Container • Random Container • Sequence Container • Blend Container • Sound SFX • Sound Voice • Audio Source • Music Switch Container • Music Playlist Container • Music Segment • Music Track <p>If this column is left blank, a default object type will be selected depending on the destination.</p> <p>Work Units can not be created at import.</p> <div data-bbox="695 913 1414 1108" style="background-color: #e0e0e0; padding: 5px; margin: 10px 0;">  <p>Note</p> <p>It is not possible to create Actor-Mixer Hierarchy objects and Interactive Music Hierarchy objects during the same import.</p> </div> <div data-bbox="695 1134 1414 1304" style="background-color: #e0e0e0; padding: 5px; margin: 10px 0;">  <p>Note</p> <p>It is not possible to create Sound SFX objects and Sound Voice objects during the same import.</p> </div>
Property[<i>PropertyName</i>] Example: Property[Voice Volume]	<p>The column header specifies which property to set. The property name is the display name, as shown in WObjects.xml, RTPC tab, Multi-Editor or List View.</p> <p>Common property names:</p> <ul style="list-style-type: none"> • Voice Volume • Voice Pitch • Voice Low-pass Filter • Voice High-pass Filter <p>Each subsequent row specifies a property value to set on the created objects.</p> <p>Enumerated values must specify the integer value associated with the enumerated name, which can be found in the .wwu files or in WObjects.xml.</p> <p>If this column is left blank, no value is set.</p>
Reference[<i>ReferenceName</i>] Example:	<p>The column header specifies which reference to set. The reference name is the full display name, as shown in WObjects.xml.</p> <p>Common reference names:</p>

Managing Media Files in Your Project

Header	Content
Reference[Output Bus]	<ul style="list-style-type: none"> • Output Bus • User Auxiliary Send 0 • User Auxiliary Send 1 • User Auxiliary Send 2 • User Auxiliary Send 3 • Conversion Settings • Effect 0 • Effect 1 • Effect 2 • Effect 3 • Attenuation • Motion Bus • MIDI Target <p>Each subsequent row specifies one of the following:</p> <ul style="list-style-type: none"> • Absolute Path: Absolute path to object in the project. <p style="margin-left: 20px;">Example: \Master-Mixer Hierarchy\Default Work Unit\Master Audio Bus \Sub Bus</p> <ul style="list-style-type: none"> • GUID: The GUID to the object, which can be found in .wwu files. <p style="margin-left: 20px;">Example:{8648567E-97C2-4e81-8853-DA0B32C463E4}</p> <div style="background-color: #f0f0f0; padding: 10px; margin: 10px 0;">  <p>Tip</p> <p>To obtain the absolute path or the GUID to an object, hold the Shift key and right-click the object to show the shortcut menu. Select Copy path(s) to clipboard or Copy GUID(s) to clipboard.</p> </div> <p>If this column is left blank, no reference is set.</p>
Notes	Defines the notes or comments for the object to be created.
Event	<p>Defines the name of an event to be created for the imported object.</p> <p>This column can contains one the following:</p> <ul style="list-style-type: none"> • Absolute path: Absolute path to an event, which starts with \Events\, followed by the work unit name. <p style="margin-left: 20px;">Example: \Events\MyWorkUnit\PlayEvent</p> <ul style="list-style-type: none"> • Relative path: Relative path to the Default Work Unit for the events. <p style="margin-left: 20px;">Example: MyVirtualFolder\PlayEvent</p> <p>To specify the action type, append @ActionName to the end of the event name. If not specified, the Play action will be used.</p> <p>Examples:</p> <ul style="list-style-type: none"> • MyEvent@Play • MyEvent@Stop • MyEvent@Pause • MyEvent@Resume • MyEvent@Break • MyEvent@Seek

Managing Media Files in Your Project

Header	Content
	<p>A Virtual Folder will be created for any path element that does not already exist at destination.</p> <p>If the specified event already exist at the destination, a new event action will be appended to the existing event.</p> <div style="background-color: #e0e0e0; padding: 10px; border: 1px solid #ccc;">  <p>Tip</p> <p>To create multiple events for a single object, you can add the Event columns multiple times.</p> </div>

Here is an example of a tab delimited worksheet that creates a random container called 'Bird' with 3 sounds.

Audio File	Object Path	Property[Voice Volume]	Reference[Output Bus]	Event
	Bird		\Master-Mixer Hierarchy\Default Work Unit\Master Audio Bus\Music	Play_Bird
C:\Sound1.wav	Bird\Sound1	-3		
C:\Sound2.wav	Bird\Sound2	-4		
C:\Sound3.wav	Bird\Sound3	-2		

To import a tab delimited file

1. In **Project Explorer**, select the import destination object.
2. Right-click the object to open the shortcut menu.
3. From shortcut menu, select **Import Audio Files...** (Shift+I).
4. Click the **Import Tab Delimited...** button.
5. Browse for the tab delimited file, and click **OK**.
6. Verify the import content in the **Audio File Importer** list.
7. Click **Import**.

Importing Voice Assets from a Text File

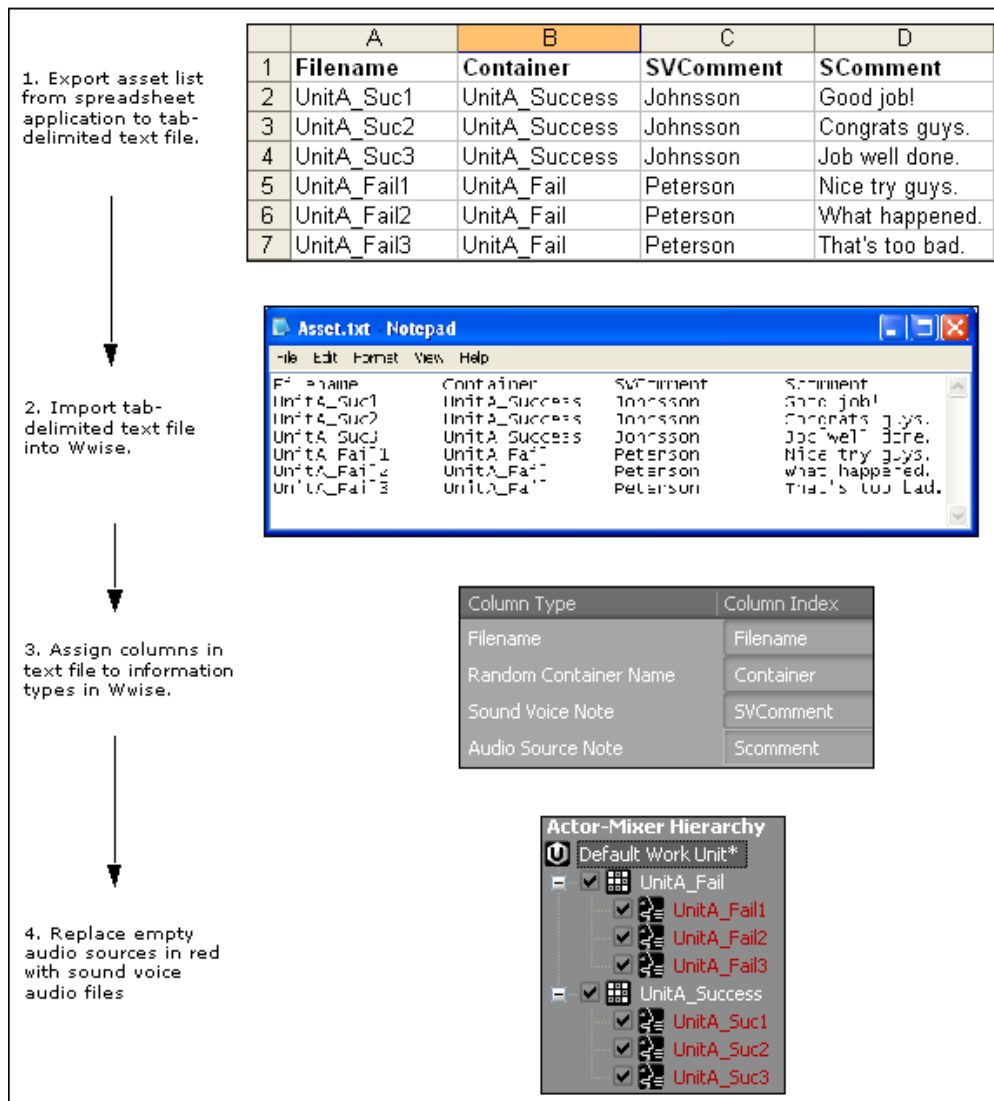
It is not uncommon to have thousands of voice assets in your game, so it is important to have an efficient way to create these voice assets in Wwise. Since these assets are generally managed in an external program, such as Microsoft Excel, Wwise allows you to recreate them in your Wwise project by importing specific information from a tab-delimited text file. This not only speeds up the creation process but also reduces the potential for error.



Note

A tab-delimited text file is a special kind of plain text file where the information is arranged into columns separated by tabs. Most spreadsheet programs, including Microsoft Excel, allow you to export the information in a worksheet to a tab-delimited text file.

The following illustration demonstrates the workflow for getting your voice asset list from a spreadsheet program into Wwise.



Although your text file may contain many different pieces of information about your assets, only the following four types of information are used by Wwise when importing the file:

- **Filenames** - This information is used to create the sound voice object and its corresponding audio source. Initially, the audio source will be empty, but it does contain a reference to an audio file with the same name. This means you can replace it in the same way you would any other audio file in Wwise. For more information on replacing media files for sound voice objects, refer to [Replacing Sound Voice Media Files](#).
- **Random container names** - This information, if included in the text file, is used to create a parent random container for the corresponding sound voice object.
- **Sound voice notes** - This information is added to the Notes field of the sound voice object. For example, you may want to add the name of the character speaking the dialogue.
- **Audio source notes** - This information is added to the Notes field of the audio source. For example, you may want to include the actual line of dialogue that is spoken.



Note

When you import voice assets into Wwise the audio sources are created in the reference language only. To create audio sources for the different language versions in your project, refer to [Importing Language Files](#).

The order and amount of information in your text file is not important, because before importing the file, you must specify which columns in your text file you want to import. The only mandatory piece of information is the filename because this name is used to create the sound voice object and the audio source. If a column in your text file has not been mapped to one of the four information types, it will be ignored by Wwise.

To help you make sure that you have correctly assigned the columns, a preview is displayed of the objects that will be created and the information that will be imported.

Sound Voice	Random Container	Sound Note	Source Note
UnitA_Suc1	UnitA_Success	Johnsson	Good job!
UnitA_Suc2	UnitA_Success	Johnsson	Congrats guys.
UnitA_Suc3	UnitA_Success	Johnsson	Job well done.
UnitA_Fail1	UnitA_Fail	Peterson	Nice try guys.
...



Caution

After you have imported the voice assets from a text file, it can't be undone.

To import voice assets from a text file:

1. Do one of the following:

From the Project menu, select **Import Voice Assets**.

Right-click a work unit, folder, or object in the Project Explorer and select **Import Voice Assets** from the menu.

The Voice Asset Importer opens.

2. Click the Browse button (...) next to the File to import text box.

The Open dialog box opens.

3. Select the text file you want to import and click **Open**.

The file's location is displayed in the File to import text box.

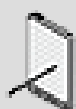
4. If there is a header row in the text file and you want to use it to assign columns, select the **Use header** check box.
5. In the Header row text box, specify which row in the text file contains the header information.
6. In the Start import at row text box, specify the row at which you want Wwise to start reading the voice asset information.
7. From the Column Index list, select the corresponding column in your text file for each of the following column types:

Filename

Random Container Name

Sound Voice Note

Audio Source Note



Note

The filename column in your text file must be assigned to the Filename column type so that the voice assets in your text file can be recreated in Wwise. All other information is optional.

8. Review the information in the Voice Assets Preview table to make sure that you correctly assigned the columns in your text file to the corresponding information types in Wwise.
9. If you want to change the location where the new sound voice assets will be created, do the following:

Click the **Browse** button beside the Import destination text box.

In the Project Explorer - Browser, select a new location in your project hierarchy.

Click **OK**.

The new path is displayed in the Import destination text box.

10. Click **Import**.

The sound voice objects are created in Wwise along with their corresponding audio sources and random containers, if any. If any problems are encountered during import, Wwise will list them in the Import Completed dialog box.



Note

The import process may take a while if you have many assets because Wwise has to verify that every asset name is unique.

Related Topics

- [Replacing Sound Voice Media Files](#)
- [Importing Media Files for Voice-Overs](#)
- [Importing Media Files for SFX](#)
- [Managing File Import Issues](#)
- [Importing Language Files](#)

Replacing Media Files

In some cases, there may be technical problems with the files that you previously imported, or you may be using temporary files as placeholders until you receive the files that you intend to use in your project. In these situations you need to replace media files that you have previously imported. You can do this in the **Audio File Importer**, or by using the Quick Replace shortcuts.

Quick Replace

The following table lists the shortcuts you can use to replace media files previously imported into Wwise.

To...	Use this shortcut:
Replace existing SFX files and create new sound objects.	Drag and drop the files.
Replace SFX files.	Alt+drag the files.
Replace existing voice files and create new sound objects.	Shift+drag the files.
Replace voice files.	Alt+Shift+drag the files. If you are replacing languages using this shortcut, make sure that the reference language is selected in the Language Selector.

Replacing SFX Files

Using the Audio File Importer, you can replace existing SFX files in your project.

To replace existing SFX media files:

1. Do one of the following:

From the Wwise menu bar, click **Project > Import Audio Files**.

Right-click an object in the **Actor-Mixer Hierarchy** and select **Import Audio Files** from the menu.

The Audio File Importer opens.

2. Click **Add** or **Add Folders**.

The Open File dialog box opens.

3. Browse to the location of the media files that you want to import.
4. Select the files, and click **Open**.

The selected files are loaded into the Audio File Importer.

5. In the Import Mode group box, select the **Replace audio files**.
6. In the Object Type group box, select **Import as Sound SFX**.
7. Click **Import**.

The Importing dialog box opens where you can view the progress of the import process.



Note

If there are errors in the media files or conflicts, the Import Conflict Manager opens. For more information on how to deal with these conflicts, refer to [Managing File Import Issues](#).

Related Topics

- [Importing Media Files](#)
- [Replacing Sound Voice Media Files](#)
- [Managing File Import Issues](#)
- [Re-Organizing Media Files within the Originals Folder](#)
- [Clearing Your Cache](#)
- [Editing Audio Files in an External Editor](#)

Replacing Sound Voice Media Files

You can replace existing voice media files in your project in the Audio File Importer in the Replace audio files mode. In this mode, the conversion settings option is not available. If you are localizing your project you will need to use the Localize Languages mode. For more information on localization, refer to [Localizing Your Project](#).

To replace voice media files in your project:

1. Do one of the following:

From the Wwise menu bar, click **Project > Import Audio Files**.

Right-click a Sound Voice or a Work Unit in the **Actor-Mixer Hierarchy** and select **Import Audio Files** from the menu.

The **Audio File Importer** opens.

2. Click **Add**.

The Open dialog box opens.

3. Browse to the location of the media files that you want to import.
4. Select the files, and click **Open**.

The selected files are loaded into the Audio File Importer.

5. In the Import Mode group box, select **Replace audio files**.
6. In the Object Type group box, select **Import as Sound Voice**.
7. From the Destination language list, select a language.



Note

The list of project languages is created in Language Manager. For more information on project languages, refer to [Managing the Languages for Your Project](#).

8. Click **Import**.

The Importing dialog box opens where you can view the progress of the import process.



Note

If there are errors in the media files or conflicts, the Import Conflict Manager opens. For more information on how to deal with these conflicts, refer to [Managing File Import Issues](#).

Related Topics

- [Importing Media Files](#)
- [Importing Voice Assets from a Text File](#)
- [Replacing SFX Files](#)
- [Managing File Import Issues](#)
- [Re-Organizing Media Files within the Originals Folder](#)
- [Clearing Your Cache](#)
- [Editing Audio Files in an External Editor](#)

Managing File Import Issues

You may encounter errors while importing media files into your Wwise project. There are two kinds of errors that may occur:

- [Managing Recoverable Errors](#) that you can resolve in the Conflict Manager.
- [Managing Non-Recoverable Errors](#) that cannot be resolved in Wwise.

Managing Recoverable Errors

When you try to import files that already exist in Wwise, and you have not selected the Replace Mode in the Audio File Importer, the Conflict Manager opens. Depending on what you are planning to do, you have three options.

- **Replace** - To replace the existing audio file with the file being imported.
- **Use Existing** - To continue to use the file that is currently linked to the audio source.

- **Cancel** - To cancel the import operation.

You can apply your options individually, or apply these options to all of the files in the error list.

To replace files with recoverable errors:

1. In the Conflict Manager window, do the following:

To replace all files, click **Replace** in the Set All To area.

To replace individual files, click **Replace** in the Error List.

2. Click **Import**.

The files in the import list are imported to the location that you have specified.

Managing Non-Recoverable Errors

When you encounter a non-recoverable error in the Conflict Manager, you must cancel the file import operation for the affected file.

While you cannot resolve non-recoverable errors in Wwise, it is helpful to review the types of non-recoverable errors that can occur. Some of these issues can be resolved outside of Wwise, and you can attempt to import the files again.

Consult the following table for a list of non-recoverable errors and suggestions about how to resolve them. For more information on the files supported by Wwise, refer to [PCM Audio File Format](#).

Cause	Suggestion
Unsupported file format	Change the format of the media files to the Waveform audio format (.wav).
Sample rate beyond specified range	Change the sample rate for the audio files to within the 1 - 96 kHz range.
Bit rate beyond specified range	Change the bit rate for the audio files to 16 bit or 24 bit.
Unsupported input channel configuration	Reduce the number of audio channels to a channel configuration between 0.1 and 7.1 or change the format of the media file to one that supports the channel configuration.
Audio file not found	For sound voice objects there is no media file for the reference language.
Audio file name is over 256 characters	Rename the media file so that it is less than 256 characters.

Related Topics

- [PCM Audio File Format](#)
- [Managing Recoverable Errors](#)

- [Clearing Your Cache](#)
- [Editing Audio Files in an External Editor](#)

Re-Organizing Media Files within the Originals Folder

As the number of source media files within your project grows, you may want to organize them into a series of sub-folders. Since Wwise needs to keep track of the full path and name of each source file, you must perform the move operation within the File Manager in Wwise.



Note

You must save your project before moving the source files in the Originals folder.

To move files between folders within the Originals directory:

1. Save your current project.
2. Do one of the following:

From the menu bar, click **Project > File Manager**.

Press **Shift+F1**.

The File Manager opens.

3. Switch to the **Source Files** tab.
4. Select the file(s) you want to move.
5. Right-click the selection and select **Move** from the menu.

The Browse For Folder dialog box opens.

6. Navigate through the folder structure and select the folder where you want to move the file.



Note

If you want to create a new folder, select the parent folder, click the Make New Folder button, and then specify a name for the folder.

7. Click **OK** to move the file to the new folder.

The Process Log dialog box opens and displays information about each file that is moved.

8. Click **OK** to close the Process Log dialog box.

Related Topics

- [Importing Media Files](#)
- [Replacing Media Files](#)
- [Managing File Import Issues](#)
- [Clearing Your Cache](#)
- [Editing Audio Files in an External Editor](#)

Clearing Your Cache

To efficiently manage your converted media files, you will need to clear your cache folder on a regular basis to remove files that are no longer used, are outdated, or are problematic. You can define the scope for this operation to selectively clear the media files in your cache. You might want to clear your cache in the following situations:

- **After you have deleted a Wwise object** - The converted media files or orphans are no longer needed, but are still associated with the objects and will remain in your cache folder until you manually clear them. In this case, you might only want to clear these.
- **A platform conversion may have produced poor quality results** - You want to start over by flushing the converted files for a particular platform.
- **The final version of language files are delivered** - You want only the newly converted files in your cache.

For each of the preceding situations you can selectively clear your cache folder based on the following:

- **Converted files** - To specify which type of files you want to clear. You can clear all converted files in your cache or only the orphan files.
- **Platforms** - To specify for which platform you want to clear the files.
- **Languages** - To specify the language files you want to clear.



Caution

Clearing your cache cannot be undone.

To clear the project cache:

1. From the Project menu, select **Clear Audio File Cache**.

The Clear Audio File Cache dialog box opens.

2. In the Audio Files group box, select one of the following options:

Only orphan files

All converted files

3. In the Platforms group box, select one of the following options:

Current platform

All platforms

4. In the Languages group box, select one of the following options for sound voice objects:

Current language

All languages

5. Click OK.

The Clearing Audio File Cache dialog box opens where you can view the clearing process.

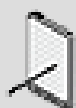
When the clearing process is successfully completed, the converted media files that you have specified are removed from your cache.

Related Topics

- [Importing Media Files](#)
- [Replacing Media Files](#)
- [Managing File Import Issues](#)
- [Re-Organizing Media Files within the Originals Folder](#)
- [Editing Audio Files in an External Editor](#)

Editing Audio Files in an External Editor

When you prepare audio files for use in game projects, you may use one or more audio editors to accomplish your tasks. After you have integrated these audio files into your project, however, you may want to edit them further. For example, you might want to cut off the end of a too-long speech, or add a note to a whistled tune. Wwise gives you the option of selecting any audio file you have already imported into your project and opening it directly in any of your favorite audio editors. In this way, you can tweak a file even after it has been integrated into your project.



Note

To use an external editor with Wwise, you must first specify each editor you want to use in the User Preferences. For more information, refer to [Selecting External Audio Editors](#).

If you have the necessary permissions, you can select imported audio files displayed in the Project Explorer and open them directly in your choice of external audio file editor. For more information on file permissions, refer to [Using Wwise with Your Source Control System](#).

To edit an audio file in an external editor:

1. On the Audio tab of the Project Explorer, right-click a sound object.

The shortcut menu is displayed.



Note

You can also right-click a source in the Contents Editor or a container or actor-mixer to edit multiple audio files at once.

2. Press *Ctrl-E* or From the shortcut menu, select **Edit in External Editor**.

The list of available editors is displayed.

3. Select the editor you want to use to edit the audio file.

The object's audio file opens in the external editor you have selected.

4. After you have finished editing the file, save it in the editor under one of the following paths:

For sounds, save under [ProjectName]\Originals\SFX\[Filename.wav]

For voices, save under [ProjectName]\Originals\Voices
\[Language]\[Filename.wav]

The edited media file is now ready to be played back in Wwise.



Caution

Do not rename the file you have edited. Otherwise, Wwise will not be able to retrieve the file.

Related Topics

- [Importing Media Files](#)
- [Replacing Media Files](#)
- [Managing File Import Issues](#)
- [Re-Organizing Media Files within the Originals Folder](#)
- [Clearing Your Cache](#)

Creating Audio and Motion Sources Using Plug-ins

Wwise's open architecture allows you to enhance your audio and motion by creating source plug-ins. These plug-ins, such as synthesizer, physical modeling, and so on, can easily be integrated into Wwise where they are used to create sound and motion objects. You can also modify the properties of the plug-in to create a wide range of sounds and motion effects.

Wwise comes with a number of different source plug-ins. The following table describes which source plug-ins are available for sound and motion objects:

Wwise Object	Source Plug-ins Available
Sound object	Wwise Audio Input Wwise External Source Wwise MP3 Input Wwise Silence Wwise Sine Wwise Tone Generator SoundSeed Air - Wind* SoundSeed Air - Woosh* * - If you plan to develop, integrate, and distribute SoundSeed Air with your game, you need to purchase a separate licence. For more information, contact the Audiokinetic sales team at: sales@audiokinetic.com .
Motion FX object (Controller)	Wwise Motion Generator

To add a source plug-in to a sound or motion object:

1. Load a sound or motion object into the Property Editor.
2. In the Contents Editor, click **Add Source**.

The Source menu is displayed with a list of source plug-ins that are available.



Note

Source plug-ins that were created for a specific platform will be unavailable in the source plug-in list for all other platforms.

3. Select the source plug-in that you want to add.

The source is added to the sound or motion object and appears as a new entry in the Contents Editor.

4. Double-click the source plug-in to display a complete list of properties in the Source Plug-in Property Editor.
5. Edit the properties as necessary.

Related Topics

- [Adding Objects to the Contents Editor](#)
- [Auditioning Objects and Sources within the Contents Editor](#)

Media File Management Tips and Best Practices

You may want to review the following sections, which provide you with a series of tips and best practices that can help you better manage your media files in your project throughout the audio development process.

Clear Cache

To save space and manage your files efficiently, you need to clear your cache regularly. You should do this in the following situations:

- After you have deleted an object.
- After you have deleted a file in the File Manager.

File Manager

Another way to keep track of your project media files is by reviewing the information on the Source Files tab of the File Manager. The File Manager keeps track of the project's Originals folder, so that you can remove any unused files if needed from your project. Removing an audio file in the File Manager does not clear the associated converted files in your audio cache however. You will need to remove them manually by selecting the Only orphan files option in the Clear Audio Cache dialog box. For more information about the File Manager, refer to [Viewing the Status of Your Project Files](#).

Chapter 7. Building Your Actor-Mixer Hierarchy

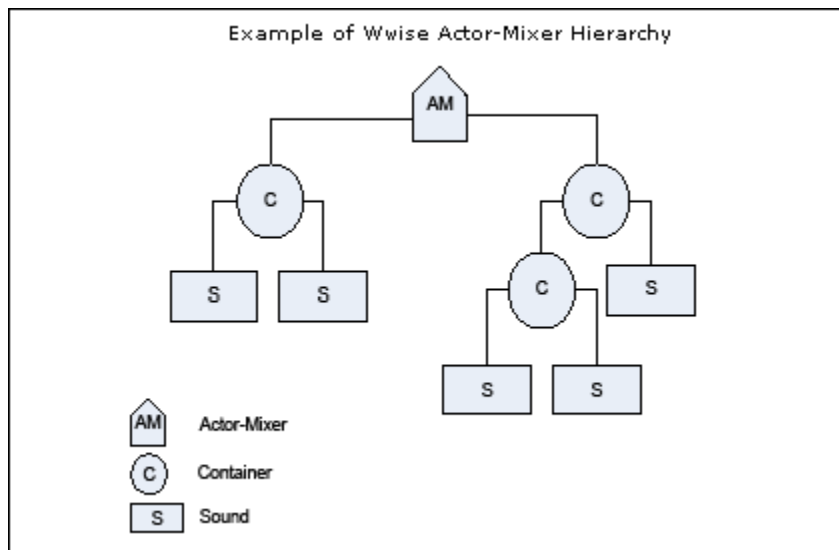
Overview	179
About Properties in the Project Hierarchy	185
Building the Actor-Mixer Hierarchy	194
Enhancing Sound and Motion by Randomizing Property Values	200
Building Actor-Mixer Hierarchy Tips and Best Practices	201

Overview

In a typical game project, you may have thousands of sound and motion assets to manage. It is important that you be able to organize these assets in the same way that you organize the many files you have on your computer. In Wwise, you can organize your project assets into special groupings that make it easier for you to work with all these assets. By grouping objects together and creating parent/child relationships, you can create a hierarchical structure. This structure not only organizes the assets in your project, but also allows you to define properties and behaviors for the group. It can be very useful to group together objects that will be sharing properties or behaviors. This gives you a great deal of power while saving you time and streamlining the development process.

You can use a combination of the following object types to group your sound and motion assets and build the **Actor-Mixer Hierarchy** in your project.

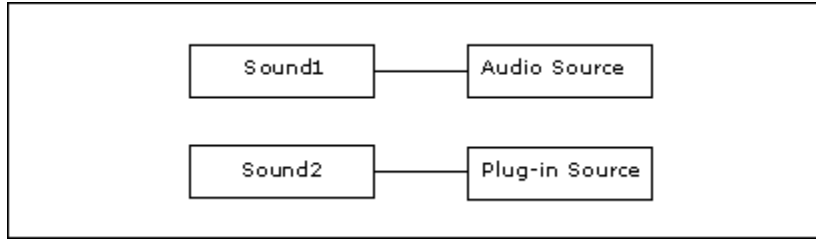
- [What is a Sound Object?](#)
- [What is a Motion FX Object?](#)
- [Types of Containers](#)
- [Using Containers and Actor-Mixers within the Actor-Mixer Hierarchy](#)



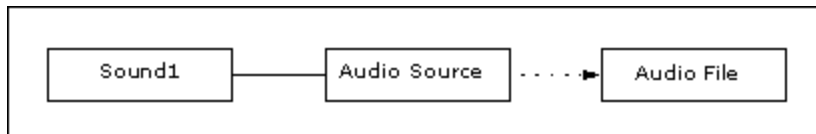
What is a Sound Object?

Sound objects are the foundation upon which your **Actor-Mixer Hierarchy** is built. They represent the individual audio assets that you have created for your project. Each sound object contains a source, which defines the actual audio content that will be played in game.

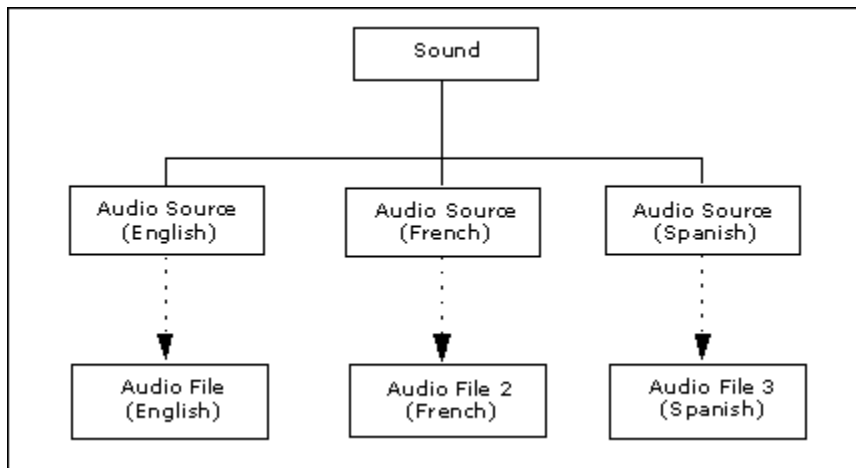
In Wwise, there are two different types of sources: an audio source and a plugin source.



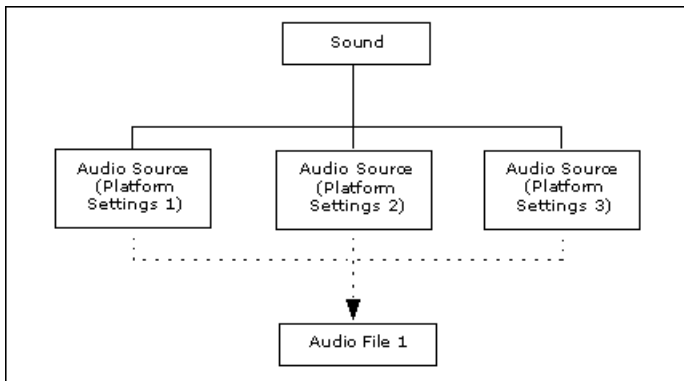
The most common type of source is the audio source. The audio source creates a separate layer between the imported audio file and the sound object. It is linked to the audio file imported into your project.



To add power and flexibility, and to accommodate multi-language and multi-platform development, Wwise allows sound objects to contain more than one source. The audio sources can link to different audio files in the case where you are using different language versions that represent the same sound.



Audio sources can also link to one audio file in the case where you are using different sample rates and other conversion settings for each platform.



What is a Motion FX Object?

Motion FX objects are similar to sound objects except they represent the individual assets that are used to generate motion in your game. Each motion FX object contains a source, which defines the actual motion content that will be played in game.

For motion FX objects there are two different types of plug-in sources:

- **Signal generator source** - Generates motion data from a variety of audio or motion generators, such as the Tone Generator and Wwise Motion Generator.

Wwise currently supports two different types of motion devices and each type supports different sources. The following table lists the signal generator sources that are available for each supported motion device.

Motion Device	Plug-in Sources Available
	Tone Generator - Motion
	Silence - Motion
Controllers	Wwise Motion Generator

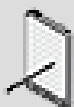
For more information about motion FX objects, refer to [Chapter 13, Managing Motion](#).

Grouping Sound and Motion Objects to Create Your Actor-Mixer Hierarchy

Different sound and motion objects can be grouped together, so that you can define the properties and behaviors for the entire group. As you group sound and motion objects together, you begin to build a hierarchical structure for your project.

In Wwise, there are two different objects that you can use to group sound and motion objects:

- [Types of Containers](#) - these are mainly used to play a group of objects according to a certain behavior, such as random, sequence, switch, and so on.
- [Using Containers and Actor-Mixers within the Actor-Mixer Hierarchy](#) - these are generally used to specify the overall properties of a group of objects, such as volume, pitch, and so on.



Note

Virtual Folders can also be used to group sound and motion objects, but there are no properties and behaviors associated with virtual folders.

Because containers and actor-mixers serve a different purpose within your game structure, the properties, behaviors, and play back possibilities for each object are different. The following table summarizes the differences between containers and actor-mixers.

Feature	Container	Actor-Mixer
Set property values	✓	✓
Define object behaviors	✓	
Play back objects	✓	

Types of Containers

Since the audio and motion in your game will require different behaviors, you can choose between several different types of containers: random, sequence, switch, and blend. Each container type has different settings that you can use to define the playback behavior of sounds and motion within the game.

Icon	Represents	Description
	Random Container	A group of one or more sounds, motion FX objects, and/or containers that are played back in a random order.
	Sequence Container	A group of one or more sounds, motion FX objects, and/or containers that are played back according to a specific order or playlist.
	Switch Container	A group of sounds, motion FX objects, and/or containers that are organized into a series of switches that correspond to the different alternatives that exist for a particular element in the game. The particular sound or motion objects that are played depend on which switch is active in the game.
	Blend Container	A group of one or more sounds, motion FX objects, and/or containers that are played back simultaneously. The different objects within this container can be grouped into blend tracks where

Icon	Represents	Description
		object properties are mapped to game parameter values using RTPCs. Crossfades can also be applied between objects within a blend track based on the value of a game parameter.

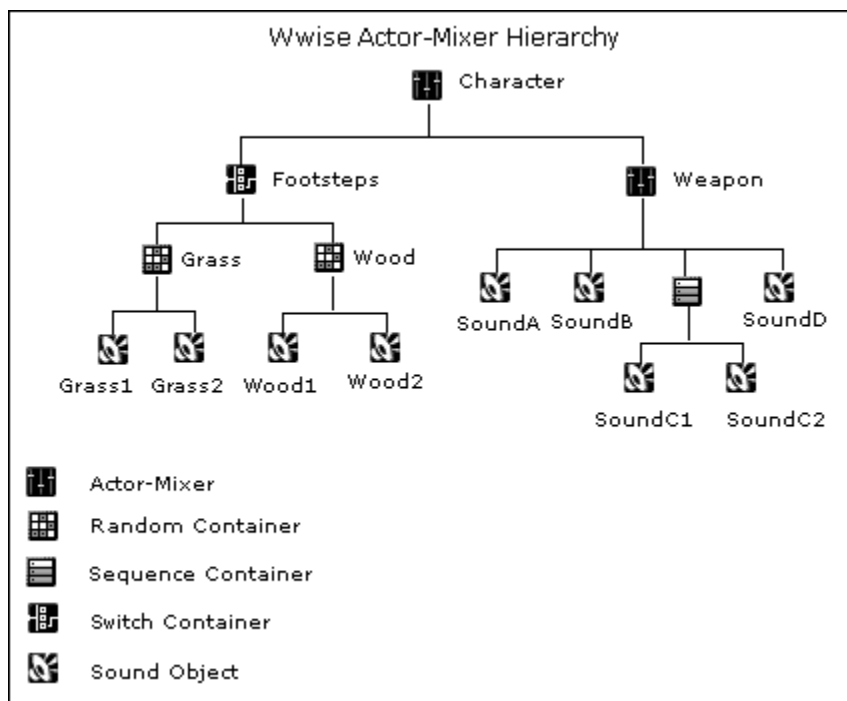
Using Containers and Actor-Mixers within the Actor-Mixer Hierarchy

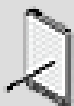
Containers and actor-mixers can both be used to group the assets within your project hierarchy, but they are applied at different levels and serve different purposes.

Containers are at the second level in the Actor-Mixer hierarchy, which means they can be both parent and child objects. You can use containers to group sounds, motion objects, and containers. By “nesting” containers within other containers, you can achieve many different results and simulate realistic behaviors.

Actor-mixers also group objects within the project hierarchy, but they sit one level above the container. This means that an actor-mixer can be the parent of a container, but not vice versa. Actor-mixers can be the parent of any number of sounds, motion objects, containers, and other actor-mixers. You can use them to group a large number of Wwise objects together in order to apply properties to the group as a whole.

The following illustration demonstrates how you can use containers and actor-mixers to group the sound assets related to one of the characters in your project.



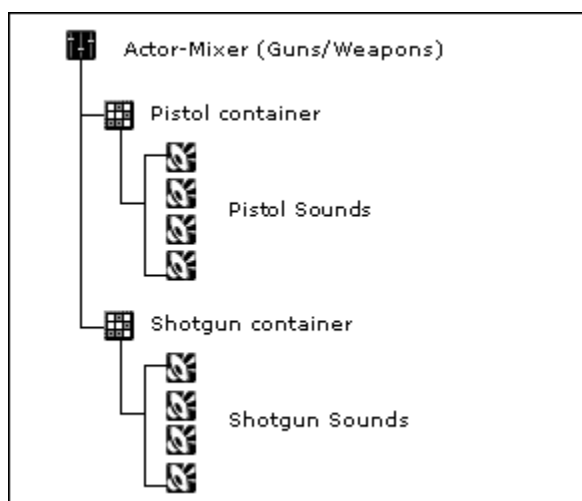


Note

A similar hierarchy of containers and actor-mixers can be created at the same level to group the motion assets in your project.

Organizing Your Assets - Example

Let's say you are working on a first-person shooter game that uses seven different weapons. You want the sounds for each weapon to have similar properties. You can group all the sounds related to a weapon into a container. Then you can group all the weapon containers into one actor-mixer so that you can control properties such as volume and pitch for all the weapons as one unit.



You can build your asset structures in the early phases of your project based on the game design. You will also need to consider other elements in your project at the same time, such as your work units, routing, game syncs and so on. Looking at your project as a whole can help you group objects together effectively. For some ideas about how to group objects, refer to [Grouping Objects in the Actor-Mixer Hierarchy](#) in the Building Actor-Mixer Hierarchy Tips and Best Practices section.

For more information on how to build your routing structures and work units, and create game syncs, refer to the following sections:

- [Chapter 5, Managing Workgroups](#)
- [Overview](#)
- [Chapter 16, Working with States](#)
- [Chapter 17, Working with Switches](#)
- [Chapter 18, Working with RTPCs](#)
- [Chapter 19, Working with Triggers](#)

- [Chapter 20, Working with States and State Groups for Dynamic Dialogue](#)

About Properties in the Project Hierarchy

Since you will be building complex structures for the audio and motion in your game, you will need to understand how the different properties are applied to objects within the hierarchy. In Wwise, the properties of an object are divided into two categories:

- **Relative properties** - Cumulative properties that are defined at each level of the hierarchy, for example pitch and volume. The sum of all these values determines the final property value. It is important to know that each relative property will be capped when it reaches either the minimum or maximum limit. The limits for each property are described below:
 - Volume: (-200 to +200) in dB
 - Pitch: (-2,400 to 2,400) in cents
 - Low-pass filter: (0 to 100) in percent
 - High-pass filter: (0 to 100) in percent

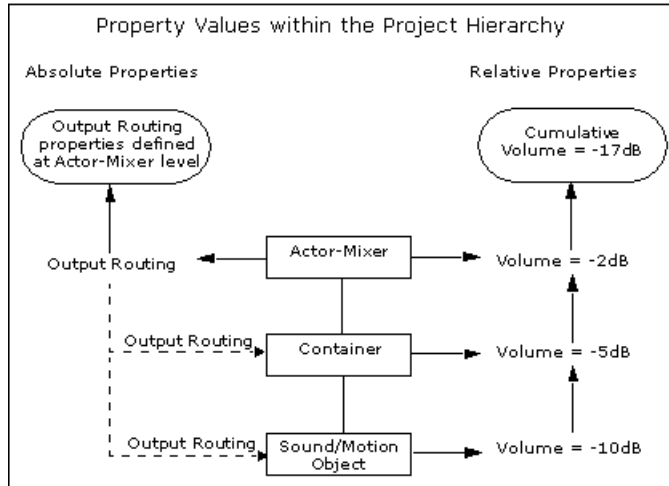


Note

Not all relative properties are available for all objects and on all platforms. For example, low pass filter properties are not available for motion objects.

- **Absolute properties** - Properties that are defined at one level, usually the highest, in the hierarchy, and then passed down to all child objects below it, for example, output routing for sound and motion objects. While you can configure these settings at the highest level in the hierarchy, they will actually be applied at each level below. An absolute property can be overridden at each level in the hierarchy.

The following illustration demonstrates how the two types of property values work within the project hierarchy.



After you have imported the assets into your project and organized them into a logical structure using the different object types, you can start defining the specific properties and behaviors of each asset.

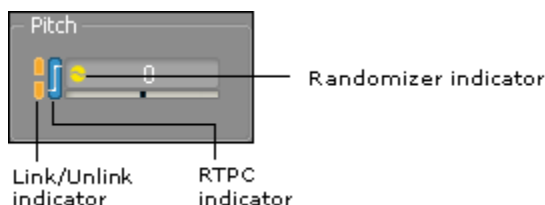
Each object in Wwise has its own set of properties and behaviors that you can use to define and differentiate it. The properties, such as volume and pitch, define the characteristics of each sound or motion object. The behaviors, such as looping and streaming, determine how the object is played back.

You can also define the properties and behaviors for all object types within the hierarchy. This allows you to control the properties of an entire group of objects, and to determine which objects within the group will be played back and in which order.








You can also apply certain effects, supported by the current platform, to the different objects to create that unique experience that you are aiming for. When applying effects at the highest level in the hierarchy, it is important to note these effects are actually applied at each level below.

Property Indicators

You may notice that certain property values in both the Property and Contents Editors have one or more indicators beside them. These indicators show whether the property value is linked to other active platforms, whether the property value is associated with a game parameter using RTPCs, and whether a Randomizer has been applied on the property value.



The following table describes the indicators that are used in each of these situations:

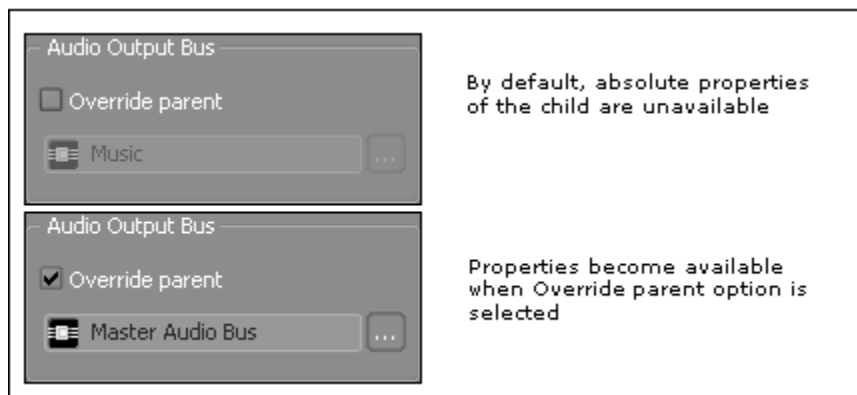
Indicator	Name	Description
	Link	A property value that is linked to the corresponding values of other active game platforms.
	Unlink	A unique property value that is not linked to the corresponding values of other active game platforms.
	Partial Unlink	The property value for the current platform is linked, but one or more corresponding values of other active platforms is unlinked.
	RTPC - On	A property value that is tied to an in-game parameter value using RTPCs.
	RTPC - Off	A property value is not tied to an in-game parameter value.
	Randomizer - On	A property value to which a Randomizer effect has been applied.
	Randomizer - Off	A property value to which no Randomizer effect has been applied.

For more information on linking/unlinking property values, using RTPCs, and randomizing property values, refer to the following sections:

- [Customizing Object Properties per Platform](#)
- [Controlling Property Values Using Game Parameters](#)
- [Enhancing Sound and Motion by Randomizing Property Values](#)

Defining Absolute Properties

Since absolute properties are automatically passed down to each of a parent's child objects, you should set them at the top-level parent object within your hierarchy. If you need to specify different properties for a particular object, you can always override the parent's properties and set new ones. By default, absolute properties are not available for child objects, but they become available when you select the Override parent option.



You can define the following absolute properties for the objects in your hierarchy:

- [Specifying the Output Routing for Sound, Music, and Motion Objects](#)
- [Specifying Output Routing for Motion](#)
- Audio to Motion Settings



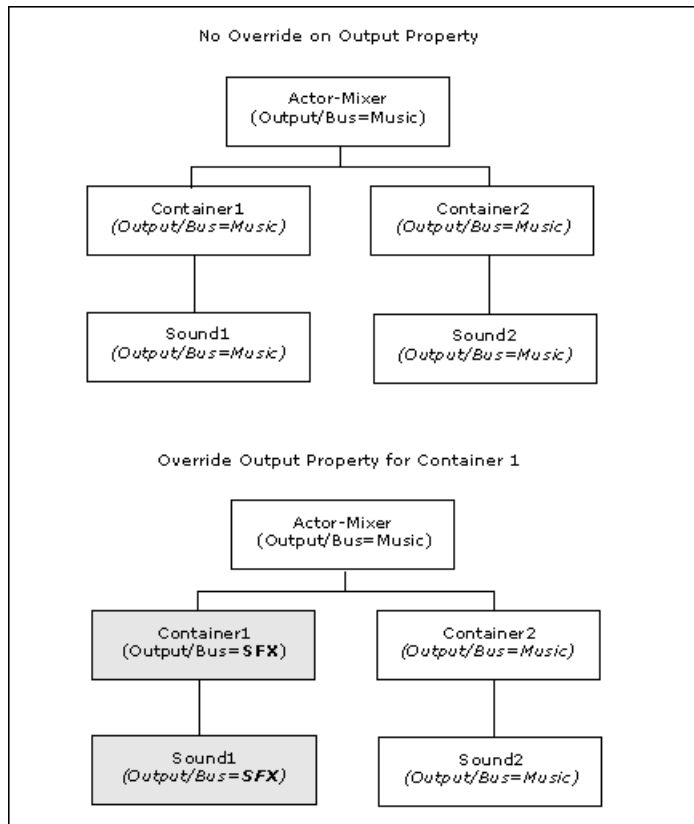
Note

The properties on the Effects, Positioning, and Advanced Settings tabs are also absolute properties, but they are discussed in a separate section. For more information about effects, positioning, and advanced properties, refer to [Chapter 12, *Managing Effects*](#), [Chapter 10, *Defining Positioning for Sound and Motion*](#), and [Chapter 11, *Managing the Priority of Sounds and Motion*](#).

Overriding Parent Properties

If you need to specify different absolute properties for a particular object, you can always override the parent's properties and set new ones. Be aware that when you override the settings for an object, you also override the property settings for its child objects as well.

The following illustration shows how the override command changes the property settings for the current object and all objects below it.

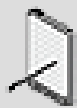


To override parent properties:

1. Load a child object into the Property Editor.
2. For any of the absolute properties, select the **Override parent** option.

The properties within the group are enabled.

3. Modify the properties as necessary.



Note

Each absolute property must be overridden separately.

Related Topics

- [Defining Absolute Properties](#)
- [Specifying the Output Routing for Sound, Music, and Motion Objects](#)
- [Defining Relative Properties \(Volume, Pitch, LPF, HPF\)](#)

Defining Relative Properties (Volume, Pitch, LPF, HPF)

Relative properties are defined for each object within your hierarchy. Unlike absolute properties, relative properties are cumulative, which means that a parent's property values are added to those of the child.

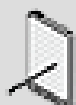
You can randomize each of the relative property values to further enhance the realism of sounds and motion within your game. For more information on randomizing property values, refer to [Enhancing Sound and Motion by Randomizing Property Values](#).

Not all relative properties are available for all objects and on all platforms. The following table lists the relative properties that are available for each object and on each platform.

Objects/Platforms	Volume		Pitch	LPF and HPF
Objects				
Sound object	✓		✓	✓
Motion object	✓		✓	
Containers (All)	✓		✓	✓
Actor-Mixer	✓		✓	✓
Music track	✓			✓
Music segment	✓			✓
Music containers (All)	✓			✓

To define the relative properties of an object:

1. Load an object into the Property Editor.
2. Type a value or drag the corresponding slider to set a value for each of the following properties:
 - *Volume*
 - *Pitch*
 - *Low-Pass Filter*
 - *High-Pass Filter*



Note

To edit the relative properties for several objects at the same time, select several objects in the **Project Explorer**, right-click the selection, and select **Multi-edit**.

Related Topics

- [Defining Absolute Properties](#)
- [Associating Low-pass Filter Values with their Corresponding Cutoff Frequencies](#)
- [Enhancing Sound and Motion by Randomizing Property Values](#)
- [Defining the Playback Behavior for Sound and Motion Objects](#)

Associating Low-pass Filter Values with their Corresponding Cutoff Frequencies

In order to save CPU usage at runtime and to use the same model for *low-pass filter* as for *volume*, *pitch*, and *LFE* (which is relative through the hierarchy), the low-pass filter property has been normalized# between 0 and 100%.

Example of LPF Values' Corresponding Cutoff Frequencies

Imagine a sound structure with a sound and a *container* where:

- LPF on Sound = 15%
- LPF on Container = 30%

In this scenario, the final LPF that will be applied at runtime is 45%, which represents a cutoff *frequency* of 1,922 Hz.

Using real# cutoff frequency values for the low-pass filter in Wwise would have looked something like this:

- LPF on Sound = 13,500 Hz
- LPF on Container = 7,000 Hz

In this example, defining the final cutoff frequency at runtime would have been unclear.

Wwise LPF Value Cutoff Frequencies

The following table provides an approximate correspondence between Wwise LPF values, ranging from 0 to 100, and the actual cutoff frequency in Hertz. This is intended as a simplified guide to the Wwise relative value system for LPF; your real cutoff frequencies may be slightly different.

LPF Value:	Cutoff Frequency (Hz):
0	20,000
1	19,567
2	19,133
3	18,700
4	18,267
5	17,833
6	17,400
7	16,967

Building Your Actor-Mixer Hierarchy

LPF Value:	Cutoff Frequency (Hz):
8	16,533
9	16,100
10	15,667
11	15,233
12	14,800
13	14,367
14	13,933
15	13,500
16	13,067
17	12,633
18	12,200
19	11,767
20	11,333
21	10,900
22	10,467
23	10,033
24	9,600
25	9,167
26	8,733
27	8,300
28	7,867
29	7,433
30	7,000
31	6,422
32	5,892
33	5,405
34	4,959
35	4,550
36	4,174
37	3,829
38	3,513
39	3,223
40	2,957
41	2,713
42	2,489
43	2,283
44	2,095
45	1,922
46	1,763
47	1,618
48	1,484







Building Your Actor-Mixer Hierarchy






LPF Value:	Cutoff Frequency (Hz):
49	1,361
50	1,249
51	1,146
52	1,051
53	964
54	885
55	812
56	745
57	683
58	627
59	575
60	528
61	484
62	444
63	407
64	374
65	343
66	315
67	289
68	265
69	243
70	223
71	204
72	188
73	172
74	158
75	145
76	133
77	122
78	112
79	103
80	94
81	86
82	79
83	73
84	67
85	61
86	56
87	51
88	47
89	43

LPF Value:	Cutoff Frequency (Hz):
90	40
91	36
92	33
93	31
94	28
95	26
96	24
97	22
98	20
99	18
100	17

Building the Actor-Mixer Hierarchy

After you have created your project, you can start creating the structure for your assets in the Audio tab of the Project Explorer. The non-music structure that will contain these assets is called the Actor-Mixer hierarchy. You can add objects to the **Actor-Mixer Hierarchy** and create relationships by grouping them at different levels in the hierarchy. The following table lists the kinds of objects that you can add.

Object	Icon	Description
Sounds		Wwise objects that represent the individual audio asset and contain the audio source. There are two kinds of sound objects:
		Sound SFX - Sound effect object.
		Sound Voice - Sound voice object.
Motion FX		Motion FX - Wwise objects that represent the assets used to generate motion in your game.
Containers		A group of objects that contain sounds, motion FX objects, or other containers that are played according to certain behaviors. You can apply properties that will affect the child objects in the container. There are four kinds of containers:
		Random Containers - A group of one or more sounds, motion FX objects, and/or containers that are played back in a random order.
		Sequence Containers - A group of one or more sounds, motion FX objects, or containers that are played according to a specific playlist.
		Blend Containers - A group of one or more sounds, motion FX objects, and/or containers that are played back simultaneously. The sounds and containers within the blend container can be grouped into blend tracks where sound properties are mapped to game parameter values using RTPCs. Crossfades can also be applied between the sounds within a blend track based on the value of a game parameter.

Object	Icon	Description
		Switch Containers - A group of one or more sounds, motion FX objects, and/or containers that are organized into a series of switches or states that correspond to the different alternatives that exist for a particular element in the game.
Actor-Mixers		High level objects into which you can group other objects such as sounds, motion FX objects, containers, and/or actor-mixers. Properties that are applied to an actor-mixer affect the properties of the objects grouped under it. You can also group objects within an actor-mixer using virtual folders.
Virtual Folders		High level elements into which you can place other objects, such as virtual folders, actor-mixers, containers, motion FX, and sounds. Virtual folders cannot be child objects for containers, motion FX, or sounds.
Work Units		High level elements that are used to divide up a project so that different people can work on the project concurrently. Work units contain the assets within your project hierarchy as well as other Wwise elements, such as states, effects, and SoundBanks.
Physical Folders		High level elements into which you can group other physical folders or work units in a project. Physical folders cannot be child objects for containers, motion FX, or sounds.

Adding Objects to the Actor-Mixer Hierarchy

Your starting place for creating your hierarchy is the work unit. If you are working alone you can begin creating your hierarchy directly from the Default Work Unit; otherwise, you will start adding objects to created work units. For more information about work units and workgroups, refer to [Chapter 5, Managing Workgroups](#).

To actually build the structure, you can do either of the following:

- Set up your project structure and then import your audio files into the structure.
- Import your audio files and organize them into a project structure that you create afterwards.

For more information on importing audio files and how they create new objects in the Actor-Mixer Hierarchy, refer to [Chapter 6, Managing Media Files in Your Project](#).

To create a child object using the Project Explorer toolbar:

1. In the **Audio** tab of the **Project Explorer**, select the work unit under which you want to create an object.

A series of icons become active in the Project Explorer toolbar.

2. From the list, click the icon that represents the object that you want to add.

The object is added to the **Actor-Mixer Hierarchy** under the selected work unit.



Note

The first time you create a motion FX object, a message box is displayed prompting you to enable one or more motion devices for your project. The motion FX object will be created regardless, but you won't be able to create any sources until a motion device is enabled. For more information on enabling motion devices, refer to [Enabling the Motion Devices for Your Project](#).

3. Replace the default name with one that best represents the object.



Note

The following characters may not be used when naming objects in Wwise: ‘:<>%*?”/\|.’

You can now start adding other objects to the hierarchy. Take the time up-front to understand the relationships between objects so that you can organize them accordingly. This will save you a great deal of time later on in the project.

To create a child object in the Actor-Mixer hierarchy:

1. In the Audio tab of the Project Explorer, right-click the work unit under which you want to create an object.
2. From the shortcut menu, select **New Child**.

The sub-menu is displayed with a list of objects that you can add.

At this level of the hierarchy, you can add any one of the following objects:

Virtual Folder

Actor-Mixer

Switch Container

Random Container

Sequence Container

Blend Container

Sound SFX

Sound Voice

Motion FX

3. From the list, click the object that you want to add.

The object is added to the **Actor-Mixer Hierarchy** under the selected work unit.



Note

The first time you create a motion FX object, a message box is displayed prompting you to enable one or more motion devices for your project. The motion FX object will be created regardless, but you won't be able to create any sources until a motion device is enabled. For more information on enabling motion devices, refer to [Enabling the Motion Devices for Your Project](#).

4. Replace the default name with one that best represents the object.



Note

The following characters may not be used when naming objects in Wwise: ':<>%*?"/\|.'

You can now start adding other objects to the hierarchy. Take the time up-front to understand the relationships between objects so that you can organize them accordingly. This will save you a great deal of time later on in the project.

Related Topics

- [Adding Parent Objects](#)
- [Copying & Pasting Objects](#)
- [Moving Objects](#)
- [Cutting/Deleting Objects](#)
- [Creating Work Units in Your Project](#)
- [Assigning Project Elements to Work Units](#)

Adding Parent Objects

After you have added the first object to your work unit you can begin adding other objects to the Actor-Mixer hierarchy, and create parent-child relationships between them. A parent object contains other objects, and after you create one, you can move existing objects under this new parent. The benefit of creating parent-child relationships is that you can change properties and define behaviors for the parent that will affect the child objects placed below it. For more information about properties in the Actor-Mixer hierarchy, refer to [About Properties in the Project Hierarchy](#).

To create a parent object using the Project Explorer toolbar:

1. In the Audio tab of the Project Explorer, select the object to which you want to add a parent object.
2. Press the **Shift** key to display the icons for the objects that can be added as a parent of the selected object.
3. From the list, click the icon that represents the object that you want to add.

The object is added to the **Actor-Mixer Hierarchy** as the parent of the selected object.

4. Replace the default name with one that best represents the object.



Note

The following characters may not be used when naming objects in Wwise: ‘:<>%*?”/|.’

To create a parent object:

1. In the Audio tab of the Project Explorer, right-click the object with which you want to create a parent relationship.
2. From the shortcut menu, select **New Parent**.

A sub-menu opens with a list of objects that you can add.

Depending on the selected object's level in the hierarchy, you have the option of adding the following as new parent objects:

Random Container

Sequence Container

Blend Container

Virtual Folder

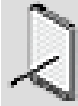
Switch Container

Actor-Mixer

3. From the list, click the object that you want to add.

The new parent object is added to the **Actor-Mixer Hierarchy** with a default name, based on its object type.

4. Replace the default name with one that best represents the object.



Note

The following characters may not be used when naming objects in Wwise: ‘:<>%*?"/\|.’

Related Topics

- [Adding Objects to the Actor-Mixer Hierarchy](#)
- [Copying & Pasting Objects](#)
- [Moving Objects](#)
- [Cutting/Deleting Objects](#)
- [Creating Work Units in Your Project](#)
- [Assigning Project Elements to Work Units](#)

Managing Objects in the Actor-Mixer Hierarchy

In the **Actor-Mixer Hierarchy** you have access to shortcut menus and the standard Windows shortcuts for renaming, cutting, copying, and pasting objects. Keep in mind the following when you make changes to the hierarchical structure.

Moving Objects

Action	Results
Moving an object	If you change an object's location in the hierarchy, the object will be affected by its new parent's properties and behaviors.
	If you move an object that is associated with an event, the object remains associated with the event.

Copying & Pasting Objects

Action	Results
Copying an object	If you copy and paste an object in a new location, it will be affected by its new parent's properties and behaviors. Its child objects will be affected as well.

Action	Results
	If you copy an object that is associated with an event, the new object will not be associated with the event.

Cutting/Deleting Objects

Action	Results
Cutting or deleting an object	If you cut or delete an object, its child objects will be deleted as well.
	The associated converted audio file is not deleted. Converted audio files that are not associated with an object are called orphan files. To delete these orphan files you need to clear your audio cache. For more information about deleting orphan audio files, refer to Clearing Your Cache .
	If you delete or cut an object associated with an event, this will result in a missing object in the event.

Enhancing Sound and Motion by Randomizing Property Values

To give each sound and motion FX object a unique flavor each time it is played in-game, you can apply Randomizers to many different property values in Wwise. Randomizers modify the property values of an object each time it is played by selecting a value from a range of values that you have specified.

When using Randomizers, you must do the following:

- [Enabling/Disabling Randomizers](#)
- [Editing Randomizer Properties](#)

Enabling/Disabling Randomizers

You can apply a Randomizer to any property that has the Randomizer icon. By default, the Randomizer is disabled, but you can easily enable it and modify the properties to create the desired effect.

To enable a randomizer:

1. Right-click the text box or slider of a property that contains the Randomizer icon.

A shortcut menu is displayed.
2. From the menu, select **Enable Randomizer**.

The Randomizer view is displayed and the Randomizer icon in the text box turns yellow.



Tip

You can also double-click the Randomizer icon to open the Randomizer view.

Related Topics

- [Enhancing Sound and Motion by Randomizing Property Values](#)
- [Editing Randomizer Properties](#)

Editing Randomizer Properties

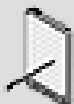
Because Randomizers only create a range of values for a particular property, they only require two properties: a Min and a Max value. These values are not absolute values, but are applied as offsets to the existing property value.

To edit the properties of a Randomizer:

1. Double-click a property's Randomizer icon.

The Randomizer view is displayed.

2. In the Min text box, type the minimum possible value that you want for the property. The Min value is applied as an offset to the existing property value.
3. In the Max text box, type the maximum possible value that you want for the property. The Max value is applied as an offset to the existing property value.



Note

To edit the properties of a Randomizer for several objects at the same time, select several objects in the Project Explorer, right-click the selection, and select Multi-edit.

Related Topics

- [Enhancing Sound and Motion by Randomizing Property Values](#)
- [Enabling/Disabling Randomizers](#)

Building Actor-Mixer Hierarchy Tips and Best Practices

You have a great deal of flexibility in creating your hierarchical structure in Wwise. Adopting a coherent strategy at the beginning of a project can save you time and effort later on. Of course, there are multiple ways to approach any

audio project; these are some concepts to consider to help you achieve the best results for your game.

Grouping Objects in the Actor-Mixer Hierarchy

Before you start building your hierarchy you need to think about the best way to organize your objects to save authoring time, but also to manage your project's memory consumption. Depending on what you are planning to do, here are some suggestions about how you can group different objects efficiently in your project.

The actor-mixer is the ultimate memory and CPU saver because some of the actor-mixer's properties, such as positioning and RTPCs, are shared by all of its child objects. So when you are considering how to organize your objects, think about grouping objects under actor-mixers to:

- Share property settings so they are processed only once.
- Limit overrides to avoid processing the overrides for each object.



Caution

Effects applied at the actor-mixer level is an efficient way to apply the effect but will not save CPU usage. When you apply an effect at the actor-mixer level, an instance of the effect is applied to all child sound or motion objects. The effect is processed for each object in real time and can therefore take up a great deal of CPU.

To optimize memory usage, consider grouping objects into actor-mixers to share the following properties:

- Positioning
- RTPCs
- States
- Randomizers

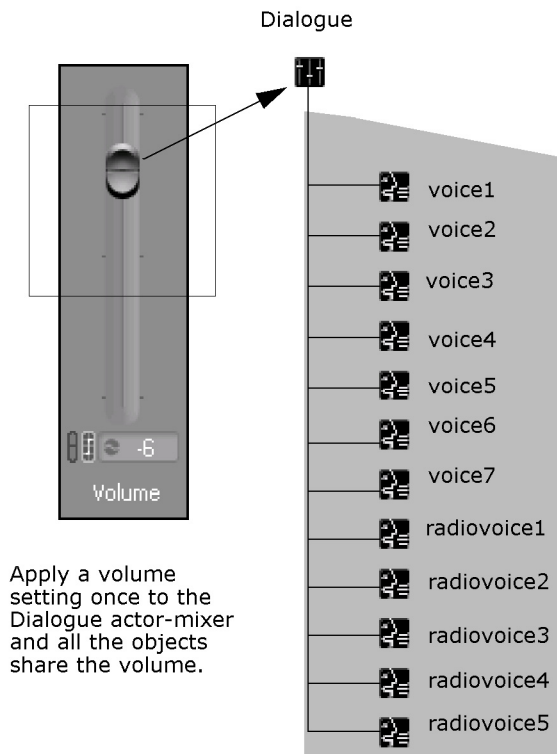
Let's say you have an actor-mixer containing 10 sounds and you want to set the sound positioning to 3D. You could set the sounds individually to 3D by using the override parent option for each sound. However, doing it this way uses 10 times more memory at run time than if you had set the actor-mixer positioning properties to 3D. Now if you wanted some of the sounds to be 2D, you would still be optimizing memory if you set the actor-mixer's positioning to 3D game-defined. In this case you would override the actor-mixer and apply 2D to the specific sounds because 2D sounds do not require additional memory.

While the actor-mixer is usually your best choice, in certain situations, you can decide to apply properties in containers to optimize memory consumption.

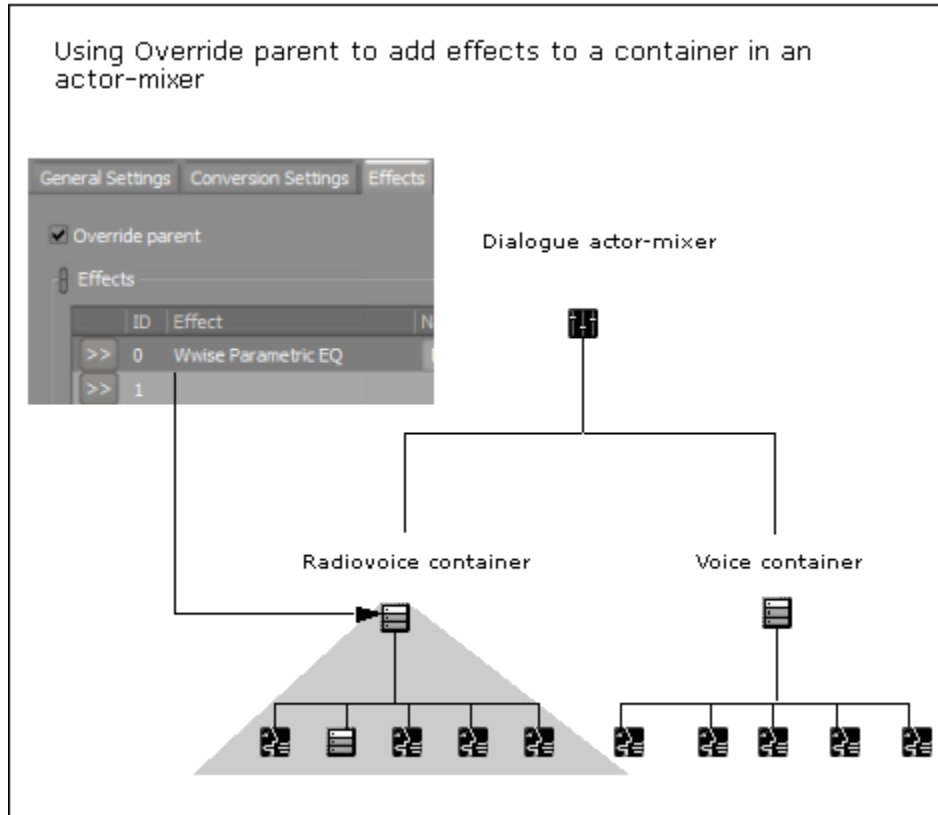
If, for example, you are only applying positioning to specific objects within a container, for example footstep sounds in a random container, you could save memory by applying the positioning properties to the container and not to the parent actor-mixer. If, however, you want all the objects in the structure to share the positioning properties, you would apply these at the actor-mixer level.

Grouping Objects - Example

Let's have a look at how you can group objects in an actor-mixer effectively using some of the concepts we have just discussed. In this example you are working with dialogue assets for your game. Some of the assets will be used as character voices and others for radio communications. You could group these under a Dialogue actor-mixer because you know that you want your dialogue to share properties such as volume, for example.



Now that you have set your volume for the actor-mixer, you have decided to add a Parametric EQ effect on the radio dialogue only. You could edit each radio voice and override the actor-mixer settings for each and apply this effect. But you could also work more efficiently by grouping all the radio voice files together in a container and then override the actor-mixer settings and add the effect on the container.



Real-time Mixing and Object Properties

When you are connected to a game or the Game Simulator you can modify the values of the following relative properties from within Wwise in real time:

- Volume
- Pitch
- Low pass filter
- Audio to Motion settings
- Center %
- RTPC values
- State and switch changes
- Triggers
- Attenuation controls
- Some audio and source effect plug-in properties.



Note

As a general rule, any property that can be mapped to a game parameter, can be modified in real time in game.

To be able to do this however, you need to load the object whose property you want to modify in the Transport Control or the Soundcaster. If the object is not loaded, your changes will not take effect because the object is not registered in the sound engine. For objects that cannot be loaded into the Transport Control such as actor-mixers, you can load a child object of the actor-mixer and this will register its parent objects in the sound engine. After you have registered the object, it will remain registered for the time that you are connected to the game.



Note

Keep in mind that if you pin an object in the Transport Control, other objects cannot be loaded until you unpin the first object. If, however, you have loaded the object in the Soundcaster, the object will be registered in the sound engine.

Relative Properties and Performance

In Wwise, certain relative properties such as pitch can affect performance on the different platforms. The mechanism for managing pitch in Wwise is based on the sample rate. Applying pitch to sounds increases CPU usage because the files must be resampled.

Chapter 8. Building the Structure of Output Busses

Overview	207
Defining the Properties of a Bus	212
Structuring a Bus Hierarchy - Example	217

Overview

A bus is a grouping of your project objects for mixing and final output, which we detail in [Chapter 29, Managing Output](#). But, well before finishing your project, it is important to understand the role of different busses and how they work within your overall project structure.

Just as you need to define the structure for your project assets to manage them strategically, you also need to organize the output for your project. By grouping output busses together in a hierarchical structure called the Master-Mixer Hierarchy, you can define the relative properties, the states, RTPCs, as well as the effects for the routing of your project.

You should spend some time considering how best to organize your project routing to streamline the mixing process. For example, you would simplify the mixing of your game audio by routing such sounds as ambient music or gunfire through corresponding busses.

To begin, you must understand the default structure in your **Master-Mixer Hierarchy**. It consists of three master buses, each with their own role and hierarchy:

- [The Master Audio Bus Hierarchy](#)
- [The Master Secondary Bus Hierarchy](#)
- [The Master Motion Bus Hierarchy](#)

The Master Audio Bus Hierarchy

The Master Audio Bus hierarchy is a structure of output busses through which the sound and music in your project are routed. It consists of three different levels of functionality:

- **Master Audio Bus** (Master Control Bus in projects from older versions of Wwise) - The top level element in the hierarchy that determines the final output of your audio. While you can move, rename, and delete busses you create, the Master Audio Bus can't be renamed or removed. You can also apply effects to the master audio bus.
- **Audio busses** - One or more optional busses that can be grouped under the Master Audio Bus to help in the organization and delivery of your sound mix. These busses can be renamed, moved, and deleted; and, for most platforms, you can also apply effects to them.
- **Auxiliary busses** - One or more optional busses that can be grouped under any auxiliary or audio bus. Like audio busses, they can be renamed, moved, copied and deleted; and, for most platforms, you can also apply effects to auxiliary (commonly referred to as "aux") busses. Sound objects anywhere in the project can be sent to auxiliary busses to adjust volume, channel configuration, positioning, and RTPC, as well as to apply effects, states, or

mixer plug-ins. Ducking, voice, and HDR mix adjustments are not possible in an auxiliary bus.



Note

Auxiliary busses cannot have audio busses as children. They can only have other auxiliary busses as children.

The following Wwise interface icons help you easily identify audio busses and auxiliary busses:

Icon	Represents
	Audio Bus
	Auxiliary Bus

By default, the sounds from the **Actor-Mixer Hierarchy** are routed through the Master Audio Bus. However, as you build your output structure, you can systematically route objects through the busses that you create.



Note

You can change the default audio routing for your project in the Default Setting dialog box. For more information about these default settings, refer to [Defining your Project Settings](#).

The Master Secondary Bus Hierarchy

The Master Secondary Bus hierarchy is a structure of audio busses used to mix the content that will go in any other outputs than the main (TV or speaker) output. For example, the most common secondary output are the game controller speakers. Like the Master Audio Bus, the Master Secondary Bus can have any number of child busses and auxiliary busses.

Sounds routed to the Secondary Bus hierarchy will be sent to the secondary output using one of the two following approaches:

- By setting the Output Bus property of a sound directly to any bus in the Secondary Bus hierarchy. This is the preferred method for sounds that are normally tied to only one secondary output instance. For example, player-initiated gun shots, tennis racket whack, PDA sounds, gameplay feedback, and so on.
- Routing a sound through any bus in the Master Audio Bus hierarchy and adding a user or game send to an auxiliary bus inside the Secondary Bus

hierarchy. This is the preferred method if the same sound is going to be heard in multiple outputs and the TV at the same time. For example: spy camera, announcements, and so on.

It is important to know that although there is only one Master Secondary Bus in the project, the bus structure will be duplicated in the game, one copy of the structure for each secondary output (game controllers, companion devices, and so on). Therefore, not all the sounds routed into the Master Secondary Bus hierarchy will be mixed together. Effective routing into which copy of the structure will depend on the Listeners and Game Object associations set up by the programmer.

To try out your sound design for Secondary Outputs, make sure to select the option "Mix Secondary Output to Main" in the Audio menu. This will make the Secondary Output sounds audible while you are using the authoring tool. Normally, these sounds would not be heard, as would be the case if you selected "Mute Secondary Output".

The Master Motion Bus Hierarchy

The Master Motion Bus hierarchy is a structure of motion busses through which the motion objects (or sound objects via audio to motion in their Motion tab) in your project are routed. It consists of two levels with different features:

- **Master Motion Bus** - The top level element in the hierarchy that determines the final motion output, typically controller vibration, of your motion or, potentially, sound objects. While you can move, rename, and delete the other busses that you create, the Master Motion Bus can't be renamed or removed.
- **Motion busses** - One or more busses for routing motion or, potentially, sound objects to a motion output device, typically a controller. They are grouped under the master motion bus and can be renamed, moved, and deleted.



Note

You can generate motion data from a sound object, including music files. For more information on generating motion from an existing sound object, refer to [Generating Motion from Existing Sounds](#).

To help you easily identify a bus in the interface, Wwise uses a unique icon to represent it.

Icon	Represents
	Motion Bus

By default, the motion FX objects from the Actor-Mixer hierarchy are routed through the Master Motion Bus. However, as you build your output structure, you can systematically route objects through the busses that you create.



Note

You can change the default motion routing for your project in the Default Setting dialog box. For more information about these default settings, refer to [Defining your Project Settings](#).

Adding Busses in the Master-Mixer Hierarchy

To create the structure for your audio routing, you can add audio busses to the Master Audio Bus. You can also create a structure for your motion routing by adding motion busses under the Master Motion Bus. After you have created a child bus under the Master Audio Bus or the Master Motion Bus, you can create any number of parent and child busses to build your routing structure.

To add a child bus in the Master-Mixer hierarchy:

1. In the Audio tab in the Project Explorer, right-click the bus for which you want to create a new child.
2. From the menu, select one of the following:

New Child > Audio Bus.

New Child > Motion Bus.

New Child > Auxiliary Bus.

The new child bus is added to the Master-Mixer hierarchy.

To create a parent bus in the Master-Mixer hierarchy:

1. In the Audio tab of the Project Explorer, right-click the bus for which you want to create a new parent.
2. From the menu, select one of the following:

New Parent > Audio Bus.

New Parent > Motion Bus.

New Parent > Auxiliary Bus.

The new parent bus is added to the Master-Mixer hierarchy.

Related Topics

- [Moving busses in the Master-Mixer Hierarchy](#)
- [Deleting Busses in the Master-Mixer Hierarchy](#)

Moving busses in the Master-Mixer Hierarchy

After you have added your busses, you may want to change the location of a bus to create different relationships between the busses.



Note

Motion busses can't be moved under the Master Audio Bus and audio busses can't be moved under the Master Motion Bus.

To move a bus to another location in the Master-Mixer hierarchy:

1. Drag the bus to the desired location.

The bus and any of its children are moved to the new location. By changing the location, the moved bus will now be affected by its new parent's properties.



Note

You cannot use cut and paste to move a bus in the Master-Mixer hierarchy.

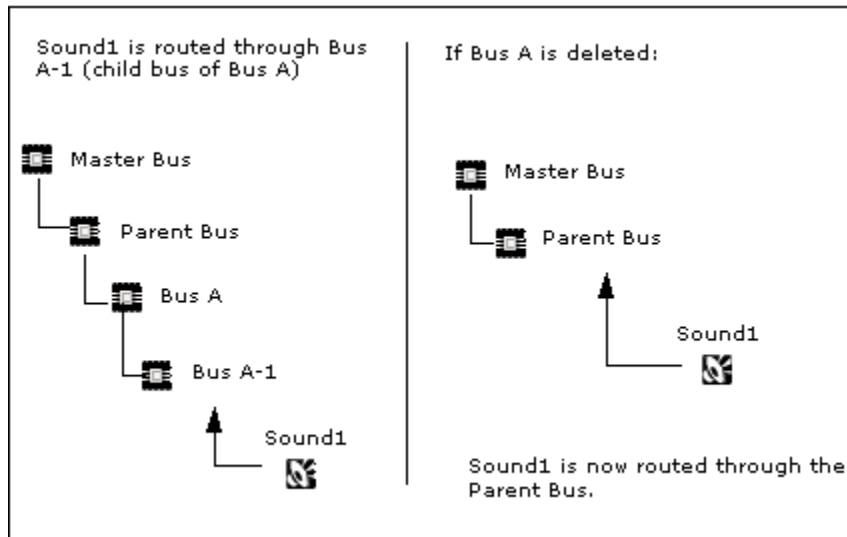
Related Topics

- [Adding Busses in the Master-Mixer Hierarchy](#)
- [Deleting Busses in the Master-Mixer Hierarchy](#)

Deleting Busses in the Master-Mixer Hierarchy

If you have created a bus by mistake or no longer need a particular bus, you can delete it. When you delete a bus, you are also deleting all of its child busses. The sounds, music, or motion objects that were routed through that bus are now reassigned to the next parent object in the hierarchy. The property overrides for the rerouted objects remain intact.

The following illustration demonstrates what happens to a sound object being routed through a bus that is deleted.



To delete a bus in the Master-Mixer hierarchy:

- Press the **Delete** key.
 - In the shortcut menu for the bus, select **Delete Selection**.
1. On the Audio tab of the Project Explorer, select the bus that you want to delete, and do one of the following:
 2. The bus and its children are deleted, and the sounds, music, or motion objects routed through the deleted busses are now routed through the parent bus.

Related Topics

- [Adding Busses in the Master-Mixer Hierarchy](#)
- [Moving busses in the Master-Mixer Hierarchy](#)

Defining the Properties of a Bus

You can use the properties of a bus to make global changes to the audio or motion in your game. When defining the properties of a bus, you can do the following:

- [Generating Motion from Existing Sounds](#)
- [Defining the Relative Properties of a Bus](#)
- [Ducking Audio and Motion Signals](#)
- [Replacing Music with a Player's Own Music](#)



Note

The LPF property is not available for motion busses.

Since the busses are the last level of control, any changes you make will affect the entire group of objects below them.

As is the case for objects, you can also apply effects, use RTPCs, assign states, and set advanced properties for busses. For more information, refer to the following sections:

- [Applying Effects to Busses](#)
- [Controlling Property Values Using Game Parameters](#)
- [Assigning States to Objects and Busses](#)
- [Chapter 11, *Managing the Priority of Sounds and Motion*](#)

Defining the Relative Properties of a Bus

Relative properties can be defined for each bus within your hierarchy. Relative properties are cumulative, which means that the property values for a bus are added to the child objects below it.

There are relative properties that you can modify for busses:

- *Bus Volume* - The attenuation directly applied in the bus.
- *Voice Volume* - The attenuation applied to audio objects playing in the bus.
- *Voice Pitch* - The playback speed of the audio objects playing in the bus.



Note

Not all properties are available for all types of busses and on all platforms; for example, LPF is not available for motion busses.

You can edit these properties for several busses at the same time in the Multi Editor. Multi-select the busses on the Audio tab of the Project Explorer and from the shortcut menu, choose **Multi-edit**. You can also tweak and fine-tune these properties in real time while connected to your game.

To define the relative properties of a bus:

1. Load a bus into the Property Editor.
2. Type a value or drag the corresponding slider to set a value to one of the relative properties.



Note

Pitch should not be applied to busses through which music objects are routed. Music objects will not be affected by any changes made to pitch in an audio bus.

Related Topics

- [Applying Effects to Busses](#)
- [Ducking Audio and Motion Signals](#)
- [Replacing Music with a Player's Own Music](#)
- [Fine-tuning your Mix using the Master Mixer Console](#)

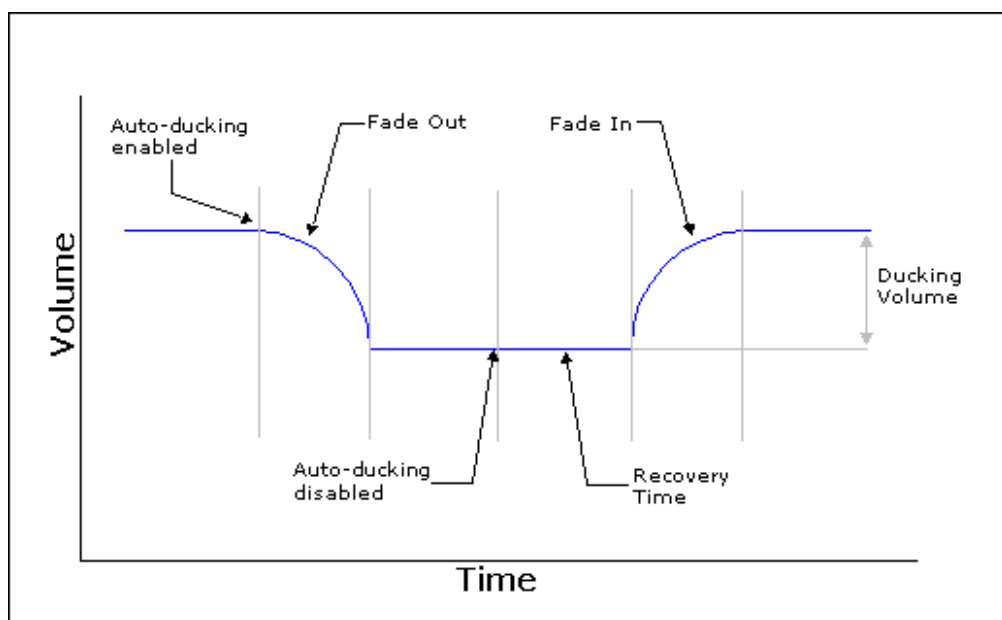
Ducking Audio and Motion Signals

At different points in your game, you will want some sounds or motion objects to be more prominent than others. For example, you may want the music to be lowered when characters are speaking in game. You can determine an audio or motion signal's importance in relation to another using ducking. Ducking allows you to automatically lower the volume level of all objects passing through one bus in order for another simultaneous bus to have more prominence.

You can control how the signals are ducked by modifying the following properties and behaviors:

- Ducking volume
- Fade out
- Fade in
- Curve shape
- Recovery time
- Maximum ducking volume

The following illustration demonstrates how and where the properties and behaviors affect the ducked signal.



To duck an audio or motion signal:

1. Load a bus into the Property Editor.
2. In the Auto-ducking group box, click **Insert**.

The Project Explorer - Browser opens with a list of busses that can be ducked.



Note

A bus can't duck itself or its direct parent.

3. From the list, select the bus that you want to duck when the current bus receives an audio or motion signal.
4. Click **OK**.

The selected bus is added to the ducking list.

5. Define the properties and behavior of the ducked bus by modifying the following settings:

Volume - The amount by which the volume of the selected bus is reduced when the current bus receives a signal.

Fade Out - The time to fade out from the original volume to the ducked volume.

Fade In - The time to fade back to the original volume.

Curve - The curve shape used to define the fade out and fade in.

6. In the Recovery time text box, specify the amount of time you want to pass from the termination of the current bus' signal to the beginning of the fade-in for the signals that were ducked.
7. In the Maximum ducking volume text box, specify The maximum amount by which the current bus volume can be attenuated when ducked by one or more busses.



Note

Ducking is not available in Auxiliary busses.

Related Topics

- [Applying Effects to Busses](#)

- [Defining the Relative Properties of a Bus](#)
- [Replacing Music with a Player's Own Music](#)
- [Fine-tuning your Mix using the Master Mixer Console](#)

Replacing Music with a Player's Own Music

The Xbox 360™, Xbox One™, PlayStation® 3, PlayStation® 4, iOS, and Android™ platforms allow their game players to replace the game music with their own. For all platforms, you must enable the **Mute for Background Music** option on all the busses you want to mute when the user's music starts. Multiple busses can be selected and this is not restricted to music busses.

To assign a bus to a platform's background music option:

1. Load an audio bus into the Property Editor.
2. Select the **Mute for background music** option.

This bus will now be muted when the user starts his music through the console's music player.

Platform-Specific details

The behavior of **Mute for background music** is slightly different on each platform. Also additional programming need to be done at the initialization of the sound engine

1. **Xbox 360:** Nothing additional to do.
2. **PS3®:** For the background music option to work on the PlayStation 3, you must also enable a switch in the sound engine initialization settings (`AkPlatformInitSettings::bBGMEEnable`).
3. **Android:** The Mute/Unmute action will occur only when the user switches from the the music player app to the game. This means that there is no "Unmute" if the user music finishes by itself.
4. **iOS:** If the `AudioSession` flag "MixOther" is set in the sound engine initialization settings, the Mute/Unmute action will occur only when the user switches from the the music player app to the game. This means that there is no "Unmute" if the user music finishes by itself. On iOS 8 and later, if the `AVAudioSessionCategoryAmbient` category is used, muting and unmuting of the game music will occur for all application audio interruptions.
5. **Xbox One and PS4:** The manufacturers added a DVR function that allows the gamer to record his games and publish them. This raised a few legal issues regarding the copyrighted music that might be part of the game audio, or user-replaceable music. While the game studio have the rights to use the music in their game, the end-user doesn't have the rights to distribute it in any form. Thus the TCRs require that background music should not

be recorded. The cost-effective solution (CPU wise) for this problem is to mix the music separately from the rest of the game. This is done using the Secondary Output feature.

The only thing needed in the Authoring tool, is to route the music objects to the **Master Secondary Bus**, or any other bus under that bus. If your game is also playing sounds on the game controllers, your project also uses this bus hierarchy for the controller sounds. Do not worry about the music being mixed with the controller sounds or vice-versa, it won't be. There is an additional step for final routing decisions made through the Listener/Game Object pairings. This is setup by the game programmer. So just as each player doesn't have the same sounds mixed in each of their controllers, the music will be sent to a different mixing structure internally. See [Understanding Secondary Outputs](#) for more information.

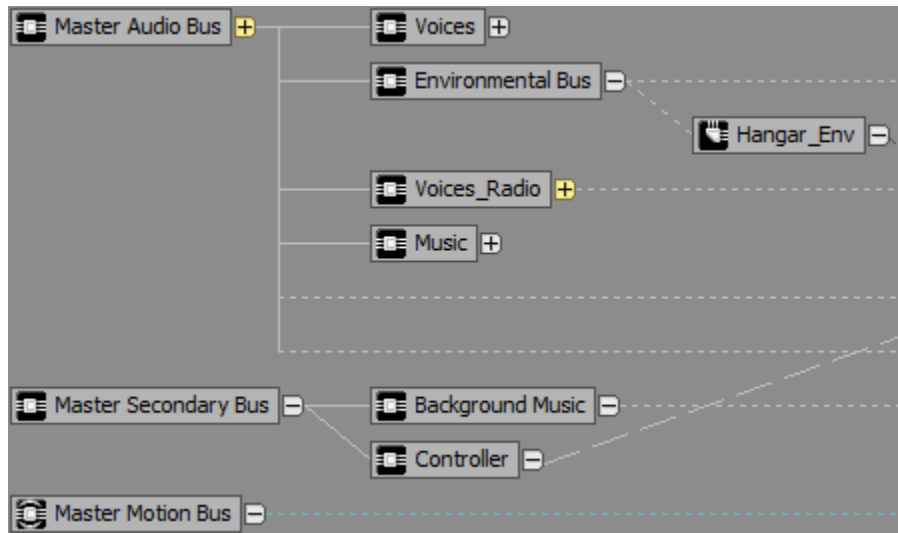
Related Topics

- [Applying Effects to Busses](#)
- [Defining the Relative Properties of a Bus](#)
- [Ducking Audio and Motion Signals](#)
- [Fine-tuning your Mix using the Master Mixer Console](#)
- [Specifying the Output Routing for Sound, Music, and Motion Objects](#)

Structuring a Bus Hierarchy - Example

Wwise provides great flexibility in structuring a bus hierarchy. This means that there is no universally correct way to organize your project's sound structure. Still, the following simple example provides overall general practices for the development of a bus structure regardless of a project's mix of assets and requirements.

In the following screenshot, drawn from the **Schematic View** (see [Chapter 41, Getting to Know the Schematic View](#) for more information on this graphical representation of your project's structure), we see a project organized under the three master busses into four audio busses, two secondary busses, and one auxiliary bus.



For the Master Audio Bus, we have the following four busses:

- **Environmental Bus** - This bus groups the various sounds the player is likely to hear based on different environmental factors, such as footsteps (the player's or other characters') on gravel, wood, or cement floors.
- **Music** - This bus groups all the music, whether while playing in certain settings in the game or moving through UI menus outside of the game.
- **Voices** - This bus groups most of the character dialogue.
- **Voices_Radio** - With a lot of dialogue in this game, and many voices needing special settings to represent their sound over a radio, we've also added this bus separate from the Voices bus. Among other ways we could have organized it, we might have set it as a child bus of the Voices bus. But, although conceptually similar, the desired sound output and the mixing it would imply made it easier for us to define it as a separate bus directly under the Master Audio Bus.

To make it easier to apply adjustments to sounds based on being located in a large airplane hangar, we also added an auxiliary bus: *Hangar_Env*. This way, when the game scene moves to the hangar, we can send sounds - manually or through a game call - to this bus where we apply a reverb reminiscent of the open echo one might hear in such an environment.

For the Master Secondary Bus, we have the following two busses:

- **Background Music** - Some platforms, like Xbox One and PS4, have a separate speaker that is ideal for outputting sound not directly related to a game. In this case, we will use it for background music that is not affected by changes in gameplay.
- **Controller** - PS4 has a controller speaker which can output sounds that the player will hear more directly or, at least, distinctively. This is ideal in such cases as the clunk of the character's head hitting the wall.



Tip

Do not send the same sound at the same time to both a secondary and an audio bus! Every system has its own level of latency. Even if it's only a couple of milliseconds, the delay in two outputs of the same sound will create a noticeable dissonance.

For the Master Motion Bus, we have no child busses. In this simple project, there aren't many motion FX, just a door sliding and a window opening, so they can both be placed under the master. However, we could also have routed sound objects to the motion hierarchy. In such a case, it could be warranted to add child motion busses even with so few motion objects in the project.



Note

You can generate motion data from a sound object, including music files. For more information on generating motion from an existing sound object, refer to [Generating Motion from Existing Sounds](#).

audio**kinetic**

Part III. Using Sounds and Motion to Enhance Gameplay



9. Defining Object Playback Behaviors	222
Overview	223
Defining the Playback Behavior for Sound and Motion Objects	223
Defining the Playback Behavior for Random/Sequence Containers.....	230
Defining the Contents and Behavior of Switch Containers	242
Defining the Contents and Behavior of the Blend Container	248
Object Playback Tips & Best Practices	259
10. Defining Positioning for Sound and Motion	261
Overview	262
Understanding Positioning in Wwise	263
Working with 2D Sound, Music, and Motion FX Objects	267
Working with 3D Sound, Music, and Motion FX Objects	269
Applying Distance-Based Attenuation	271
Defining Spatial Positioning Using Animation Paths	287
Routing Audio Signals to the Center Speaker	298
Positioning Tips and Best Practices	299
Understanding Channel Configurations	306
Speakers vs Headphones Panning Rules	308
11. Managing the Priority of Sounds and Motion	310
Overview	311
Understanding How Wwise Prioritizes Sounds and Motion Objects....	313
Limiting Object Playback Instances	316
Defining Playback Priority	320
Managing Low-Volume Sounds and Motion Objects	323
Priority Tips and Best Practices	325
12. Managing Effects	329
Overview	330
Using Effects	330
Using Effects to Implement Environment Acoustics	340
Effects Tips and Best Practices	344
13. Managing Motion	346
Overview	347
Understanding How Motion Works in Wwise	347
Creating Motion for Your Game	350
Building an Output Structure for Motion	358
Motion Tips and Best Practices	359

Chapter 9. Defining Object Playback Behaviors

Overview	223
Defining the Playback Behavior for Sound and Motion Objects	223
Defining the Playback Behavior for Random/Sequence Containers	230
Defining the Contents and Behavior of Switch Containers	242
Defining the Contents and Behavior of the Blend Container	248
Object Playback Tips & Best Practices	259

Overview

Different situations within a game will require different kinds of audio or motion playback. To accommodate these different scenarios, Wwise allows you to define the playback behaviors of each individual object in the Actor-Mixer hierarchy. However, to give you additional flexibility and control over sound and motion playback, Wwise also allows you to group objects into different types of containers:

- [Creating a Random Container](#)
- [Creating a Sequence Container](#)
- [Defining the Contents and Behavior of Switch Containers](#)
- [Creating a Blend Container](#)

The type of container you choose will determine how the objects within the container will be played back. For example, random containers play back the contents of the container randomly, sequence containers play back the contents of the container according to a playlist, switch containers play back the contents of the container based on the current switch or state within the game, and blend containers play back all objects within the container simultaneously while using blends and RTPCs.

Using a combination of the different containers, you can manage the various playback scenarios for your game efficiently and creatively, which ultimately enhances the experience of your game.

Defining the Playback Behavior for Sound and Motion Objects

Each sound and motion FX object within your project has a certain set of behaviors. The behaviors determine how many times an object will play each time it is called and whether the object is stored in memory or streamed directly from the DVD, CD, or hard drive.

You can define the following behaviors for sound and motion FX objects:

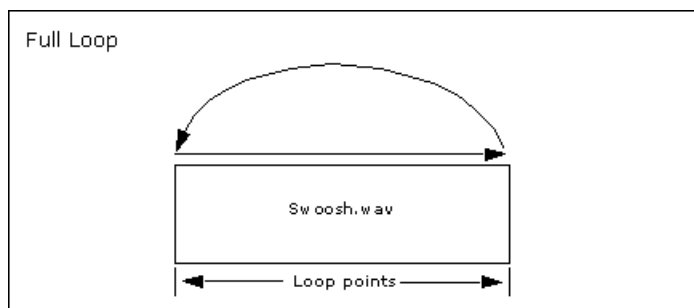
- [Creating a Loop](#)
- [Streaming Your Media](#)

Creating a Loop

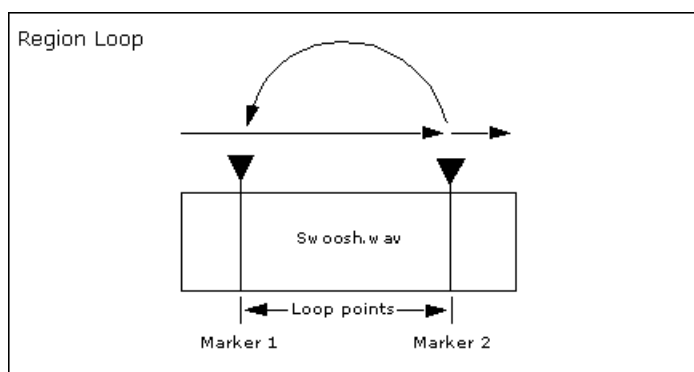
By default, sound and motion objects play once from beginning to end. You may, however, want a sound or motion FX object to play more than once. In this case, you can create a loop. When you decide to loop a sound or motion FX object, you must define the number of times it will be looped. You have

the option to loop the sound or motion FX object indefinitely or to specify the number of times it will be looped.

By default, the entire object is looped, but if loop region markers have been added to the audio file, you can loop one part of a sound or motion FX object. In a full loop, the entire object plays from beginning to end. When the object reaches its end, it returns back to the beginning to play again. It will play repeatedly until it reaches the number of loops specified.



In a region loop, only the region defined by the markers is looped. The object plays from the beginning until it reaches the end of the looped region. It then returns to the beginning of the marked region and plays the looped region repeatedly until it reaches the number loops specified. After the looping is finished, the last portion of the sound or motion FX object is played. If the looping is infinite, the last portion of the object is never played. Wwise supports only one loop region, so make sure that only one region is defined in the audio file.



Note

Region markers (or start/end samples in Audition) must be created in a third-party application, such as Adobe® Audition® and Sound Forge®. Other third-party applications may work, but they are not officially supported by Wwise.

To loop a sound:

1. Load a sound or motion FX object into the Property Editor.
2. Select the **Loop** option.

The Loop options become available.

3. Select one of the following options:

Infinite to specify that the entire object or defined loop region in the file will be repeated indefinitely.

No. of Loops to specify a particular number of times that the entire object or defined loop region will be played.

4. If you selected the No. of Loops option, type the number of times you want the object or defined loop region to be played.

Related Topics

- [Defining the Playback Behavior for Sound and Motion Objects](#)
- [Streaming Your Media](#)

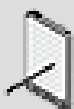
Looping Compressed Audio Files

Since compressed file formats, such as XMA and ADPCM require that file lengths and region markers fall on sample boundaries, the zero-based samples are padded during the conversion process. This extra padding can cause problems when sounds are looped. To avoid these problems, Wwise prompts you to re-convert looped sounds to ensure that:

- Padding does not occur at the end of the file.
- Loop region markers are aligned with sample boundaries.

Wwise uses a pitch shift during the re-convert process to ensure that the files meet the requirements of the compression format. The loops remain sample accurate and the sample rate of the file is not changed.

You may also notice that looped XMA files sound somewhat differently on Windows and the Xbox 360. This occurs because the Xbox 360 uses the hardware to decode the file, which means it can apply short fades between loop points. On the PC, however, a separate WAV file is generated using the XMA encoder.



Note

Any loop markers in the audio file that are shorter than the sample boundaries are removed when converted in Wwise.

Using the Source Editor

The Source Editor can be used to edit the following types of objects:

- Audio Source (encapsulate a WAV file)
- Source Plug-ins (Tone Generator, Woosh, Wind, etc.)

Editing Audio Sources

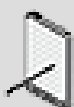
In the source editor the following operations can be applied to WAV files non-destructively:

- Trim the beginning or end of the source
- Apply fade-in or fade-out
- Set loop points
- Set a crossfade for smooth looping

All these operations are processed offline and not in real-time in the game. They are stored in the project's work units, are undoable, and reversible.

Trimming the Content of Audio Sources

Trimming the content in the WAV file saves space in WAV data. It can be used to remove silence at the beginning or the end of WAV files. Trimming the content is a non-destructive operation that occurs during conversion of the source material.



Note

The editing functions such as trimming, looping, and fades are not available for Audio Source objects in the interactive music hierarchy, because these operations are available as part of the Music Segment and Music Clip objects.

To trim the source:

1. Drag the **Trim Start** (blue square handle located on the lower-left corner of the waveform) to the right.
2. Drag the **Trim End** (blue square handle located on the lower-right corner of the waveform) to the left.

Looping Audio Sources

Loop points are used when the Audio Source is contained by a Sound object that has the **Loop** property enabled. If loop points are present in the WAV file,

they are used by default until the loop points in the editor are moved. As soon as the loop points are modified in the editor, the loop points from the file are no longer used.

The loop points are only used in the following conditions:

- The **Override file loop point** is enabled
- The parent Sound object is looping

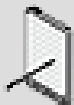


Note

Setting or modifying the loop points in the **Source Editor** does not modify the original WAV file. This is a non-destructive operation. The loop points are stored in the project.

Editing the Crossfade Duration

A crossfade duration can be defined on the audio source to obtain a smoother looping behavior in case a clicking sound is heard during the loop.



Note

To avoid looping clicks and pops, it is always recommended to position loop points at a zero crossing positions of the PCM data.

To adjust the loop points, with a crossfade:

1. Drag the **Loop Start** (green handle, located on the top-left corner of the waveform) to the right.
2. Drag the **Loop End** (Red handle, located on the top-right corner of the waveform) to the left.
3. Define a crossfade duration.

Editing Fade-In and Fade-Out

A fade-in and fade-out can be created in the source editor by dragging the fade handles (blue triangle). Fades are applied to the converted file, and do not require additional processing during the playback of the source.

To create a fade-in and a fade-out

1. Drag the **Fade-in** handle (blue triangle handle, located on the top-left corner of the waveform) to the right.

2. Drag the **Fade-out** handle (blue triangle handle, located on the top-right corner of the waveform) to the left.

Resolving Audio Source Integrity Issues

When editing audio source properties or editing a WAV file using an external editor, the selected operations on the audio source may become non applicable or cannot be fully honored.

Example of a scenario causing an integrity issue:

1. Import a WAV file in Wwise, creating a Sound and an Audio Source object.
2. In the **Source Editor**, trim the end of the WAV file.
3. In an external WAV editor, delete a region at end of the WAV file that is larger than the end trim position.

In Wwise the trim operation is on a portion of the WAV file that no longer exists. This is identified as an Audio Source Integrity issue.

Identifying Audio Source Integrity Issues

Audio source integrity issues are identified in the following Wwise locations:

- In the **Source Editor**, with a Yellow Triangle with an exclamation mark
- In the **Integrity Report**, when running the Audio Source tests
- During Audio Source conversion performed when generating SoundBanks

Repairing Audio Source Integrity Issues

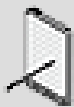
To repair an audio source integrity issue, do one of the following:

- Manually repair the audio source by addressing the unsupported operation.
- Click the yellow triangle in the **Audio Source Editor**.
- Right-click an Audio Source or Sound object (from the Integrity Report, or any other location listing audio sources), and select **Repair Audio Source Integrity Issues**.

Streaming Your Media

You can determine which sounds, music tracks and motion FX objects will be played from memory and which ones will be streamed from a hard drive, CD, or DVD. When media is streamed from the disk or hard drive, you also have the option to avoid any playback delays by creating a small buffer that covers the latency time required to fetch the rest of the file. You can specify the size of the buffer so that it meets the requirements of the different media sources, such as hard drive, CD, and DVD. Unlike the rest of the streamed file, this small buffer

is actually stored within a SoundBank and loaded with the SoundBank into memory at the appropriate point in the game.



Note

Audio playback in Wwise is always streamed regardless of whether the streaming options have been selected. The Stream settings, therefore, only apply when generating SoundBanks, and playing back from a remote platform or game.

To stream a sound:

1. Load a sound or motion FX object into the Property Editor.
2. Select the **Stream** option.

The Stream options become available.

3. Select the **Zero Latency** option to have no delay from the time the object is triggered to when it is actually played. To achieve zero latency, a certain portion of the beginning of the sound or motion data file must be stored in memory to cover the latency time required to fetch the rest of the file from the media.
4. In the Prefetch length text box, type the number of milliseconds of the sound or motion data file that you want to store in memory.

Related Topics

- [Defining the Playback Behavior for Sound and Motion Objects](#)
- [Creating a Loop](#)

Streaming Your Media from RSX™ local memory on the PlayStation®3

Lots of titles use the RSX™ local memory to cache audio data. Although RSX local memory is RAM, it cannot be read by the Cell Broadband Engine™ directly without degrading performance dramatically. Thus, media stored in RSX has to be streamed as if it was a regular storage device. You can determine which sounds, music tracks and motion FX objects should be played from RSX. Checking the RSX™ checkbox has the same effect than unlinking and checking the Stream checkbox for the PlayStation®3 only. The only difference is that they are annotated with a special flag in the description file generated along with soundbanks, SoundBanksInfo.xml. In order to play these files from RSX, you also need to use the File Packager as a post-soundbank generation step, and initialize the Wwise sound engine in your game with an RSX device.

To stream a sound from RSX™ local memory:

1. Load a sound or motion FX object into the Property Editor.
2. Ensure that the current platform is PlayStation®3.
3. Select the RSX™ option.

The Stream options become available. Use them as you would with any other streaming device. See [Streaming Your Media](#) above.

4. In the SoundBank settings, the post-SoundBank-generation step for the PlayStation®3 should launch the File Packager, in order to create packages usable by the RSX streaming device.

The game code should create an RSX device. This device loads these packages into RSX local memory and streams them from there. Refer to the [Wwise SDK documentation](#) for more details.

File packaging and run-time RSX streaming can be entirely rewritten by your own asset deployment technology.

Related Topics

- [Defining the Playback Behavior for Sound and Motion Objects](#)
- [Streaming Your Media](#)

Defining the Playback Behavior for Random/Sequence Containers

Random and sequence containers give you varied options for organizing audio and motion playback in your game. Random containers allow you to create interesting soundscapes and enhance the players experience by randomly playing from a selection of sounds or motion FX objects, while sequence containers allow you to consistently play back sounds and motion FX objects in a designated order.

You can perform the following tasks when working with random and sequence containers:

- [Creating a Random Container](#)
- [Creating a Sequence Container](#)
- [Creating a Playlist](#)

Creating a Random Container

For all situations where you want a series of sounds or motion FX objects to be played back randomly, you can use a random container. For example, you might

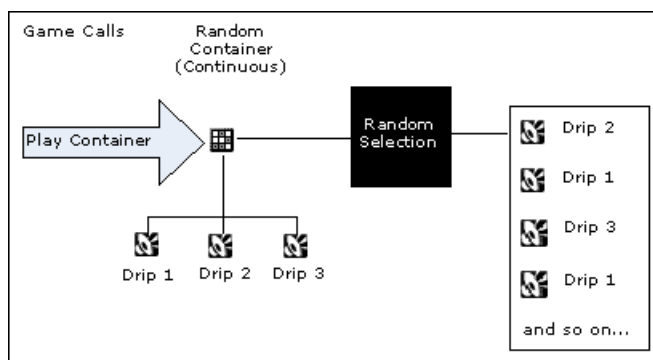
want to use a random container for all the attack, hit, and reaction sounds in a fighting game.

In Wwise, random can mean either a standard random selection, where each object within the container has an equal chance of being selected for playback, or a shuffle selection, where objects are removed from the selection pool after they have been played.

You can also assign a weight value to each of the objects within a random container. The weight value can increase or decrease the likelihood that an object is selected for playback. The weight value set for a particular object is relative to the sum of all weights in the container. This basically means that the number assigned for each object represents the number of chances the object will be selected for playback out of the summed total. For example, if there are two objects in the container with weight values of 1 and 100, the first object will have a 1/101 chance of being played and the second object will have a 100/101 chance of being played.

Using Random Containers - Example

Let's say one of the environments in your game is a cave. You want to have the sound of water dripping in the background to give some ambience to the cave environment. In this case, you can create a random container that groups all the different water dripping sounds. Since you would want the sounds to be played continuously while the character is in the cave, you would set the play mode of the container to Continuous with infinite looping. By playing the limited number of sounds randomly, you add a sense of realism.



To create a random container:

1. In the Project Explorer, right-click any one of the following objects in the Actor-Mixer hierarchy:

Work unit

Virtual Folder

Actor-Mixer

Random Container

Sequence Container

Switch Container

Blend Container

2. From the shortcut menu, select **New Child > Random Container**.

A new random container is created and highlighted in the Actor-Mixer hierarchy.

3. Type a name for the random container and press **Enter**.
4. Double-click the new random container to open its corresponding Property and Contents editors.
5. Populate the random container by dragging objects from the Project Explorer to the Contents Editor.
6. In the Random group box, select one of the following options:

Standard to keep the pool of objects intact. After an object is played, it is not removed from the possible list of objects that can be played and can therefore be repeated.

Shuffle to remove objects from the pool after they have been played. This option avoids repetition until all objects have been played.

7. To avoid objects being repeated one after the other, select the **Avoid repeating last x played** option.

The behavior of this option is affected by whether you are in Standard or Shuffle mode.

In Standard mode, the object played is selected completely randomly, but the last x objects played are excluded from the list.

In Shuffle mode, when the list is reset, the last x objects played will be excluded from the list.

8. In the Avoid repeating last x played text box, type the number of objects that must be played before an object can be repeated.
9. In the Contents Editor, assign a weight to each of the objects within the container.

The weight helps to prioritize certain objects over others. The number you assign for each object represents the number of chances the object will be selected for playback out of the summed total.

Related Topics

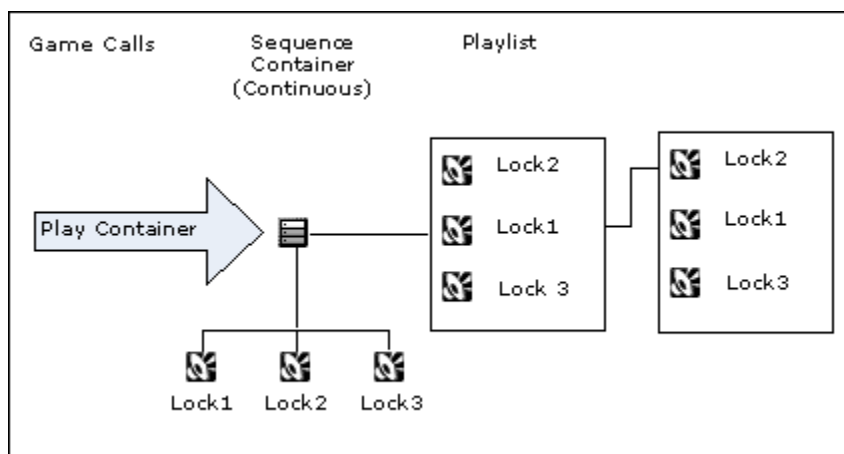
- [Defining the Scope of a Container](#)
- [Defining How Objects Within a Container are Played](#)
- [Creating a Sequence Container](#)

Creating a Sequence Container

For all situations where you want a series of sounds or motion FX objects to be played back in a particular order, you can use a sequence container. The sequence container plays back the objects within the container according to a specified playlist. For example, you can use a sequence container for delivering a character's dialogue. You would want to specify an order to the dialogue so that the character doesn't say “goodbye” before saying “hello”.

Using Sequence Containers - Example

Let's say you are creating a first person shooter game. At one point in the game, the player must push a button to open a huge steel door with many unlocking mechanisms. In this case, you can group all the unlocking sounds into a sequence container. You would then create a playlist to arrange the sounds in a logical order. You would set the play mode of the container to Continuous so that the unlocking sounds play one after the other as the door is being unlocked.



To create a sequence container:

1. In the Project Explorer, right-click any one of the following objects in the Actor-Mixer hierarchy:

Work unit

Virtual Folder

Actor-Mixer

Random Container

Sequence Container

Switch Container

Blend Container

2. From the shortcut menu, select **New Child > Sequence Container**.

A new sequence container is created and highlighted in the Actor-Mixer hierarchy.

3. Type a name for the sequence container and press **Enter**.
4. Double-click the new sequence container to open its corresponding Property and Contents editors.
5. Populate the sequence container by dragging objects from the Project Explorer to the Contents Editor.
6. To define the At end of playlist behavior, select one of the following options:

Restart to play the list in its original order, from start to finish, after the last object in the playlist is played.

Play in reverse order to play the list in reverse order, from last to first, after the last object in the playlist is played.



Tip

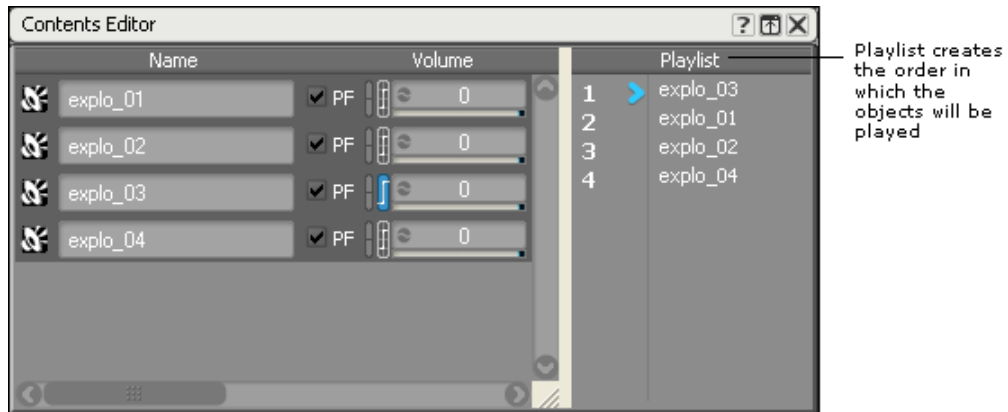
You can also create a parent sequence container for several sounds, motion FX objects, and other containers by selecting the objects in the Project Explorer and then right-clicking and selecting **New Parent > Sequence Container**.

Related Topics

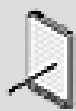
- [Creating a Playlist](#)
- [Defining How Objects Within a Container are Played](#)
- [Defining the Scope of a Container](#)
- [Creating a Random Container](#)

Creating a Playlist

When you create a sequence container, a Playlist pane is added to the Contents Editor. You can create a playlist so that objects within the container are played back in a particular order.



You can experiment and fine-tune your playlist by adding, removing, and re-ordering objects until it is just right. You can play back a playlist at any point in the creation process. For more information on playing a playlist, refer to [Playing a Playlist](#).



Note

A sequence container can't be played until a playlist has been created.

Adding and Removing Objects from a Playlist

You can easily create your playlists by adding and removing sounds, motion FX objects, and other containers to the Playlist pane of the Contents Editor. To help speed up the process, you can select several objects and then add or remove them all at the same time.

In most cases, you will be using the objects within the sequence container to create the playlist. You can, however, drag objects directly from the Audio tab of the Project Explorer to the playlist. In this case, the objects are moved from their current location to the sequence container. If you want to create a copy of the object instead of moving it, you can Ctrl+drag the object from the Audio tab in the Project Explorer to the playlist.

To add/remove objects from the playlist:

1. Load a sequence container into the Property Editor.

The objects within the container are displayed in the Contents Editor.

2. To add an object to the playlist, drag an object from the Contents pane to the Playlist pane.

The object is added to the playlist.

3. To remove an object from the playlist, click the object that you want to remove in the Playlist pane.
4. Press the **Delete** key.

The object is removed from the playlist, but still remains in the Contents pane.

Related Topics

- [Creating a Playlist](#)
- [Re-ordering Objects in the Playlist](#)
- [Playing a Playlist](#)

Re-ordering Objects in the Playlist

While you are creating your playlist, you can re-order the objects to see how they sound or feel in different orders. If your playlist has containers, you cannot re-order the objects that are nested within it.

To re-order objects in the playlist:

1. In the Playlist pane of the Contents Editor, select the object or objects that you want to move.
2. Drag the objects to the new location in the playlist.

The objects are moved to the new location.

Related Topics

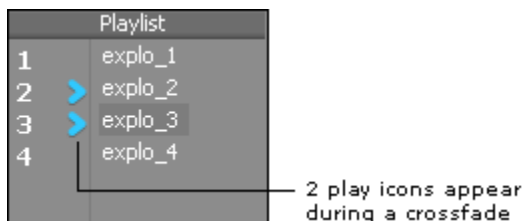
- [Creating a Playlist](#)
- [Adding and Removing Objects from a Playlist](#)
- [Playing a Playlist](#)

Playing a Playlist

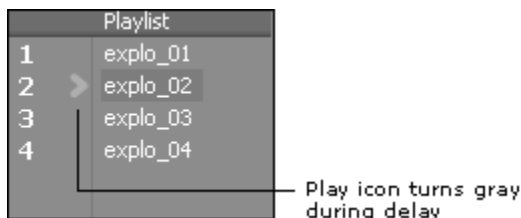
Before you can play a sequence container, you must create a playlist. A blue play icon shows which object in the playlist is playing at any one time.



If you are using a crossfade between objects, two play icons will appear in the playlist during the crossfade.



If you are using a delay between objects, the play icon turns from blue to gray during the delay.



To play a playlist:

1. Load a sequence container into the Property Editor.
2. In the Transport Control, click the **Play** icon.

Wwise plays back the playlist in the order specified.

Related Topics

- [Creating a Playlist](#)
- [Adding and Removing Objects from a Playlist](#)
- [Re-ordering Objects in the Playlist](#)

Defining How Objects Within a Container are Played

Since both random and sequence containers consist of more than one object, you must specify a play mode. The following play modes are available in Wwise:

- **Step** - Plays only one object in the container each time the container is played.
- **Continuous** - Plays the complete list of objects in the container each time the container is played.

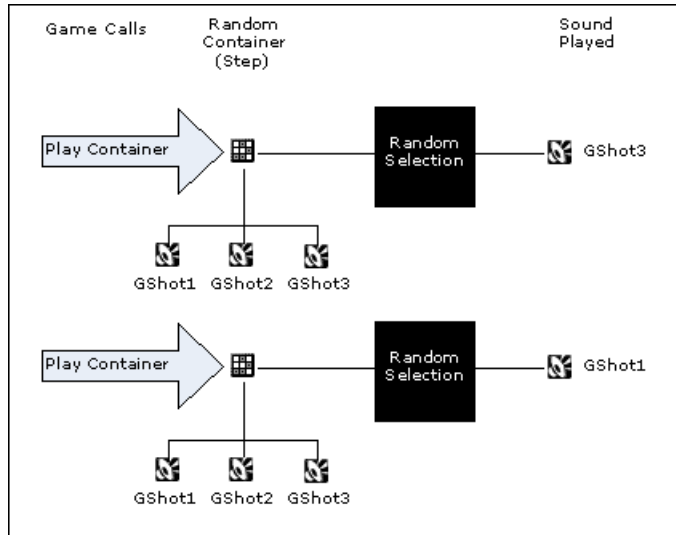
If you set the play mode to Continuous, you also have the option to loop the sounds or motion FX objects and create transitions between the various objects within the container.

Playing One Object Within the Container

In certain situations, you will only want one sound or motion FX object within the container to be played each time it is called. For example, each time a handgun is fired you may only want one sound to be played or each time a

character speaks you may only want them to deliver one line of dialogue. In these situations, you would set the play mode of the container to Step.

The following illustration shows what happens when a handgun is fired and the sounds for that handgun are organized into a random container in step mode.



To play one object within the container:

1. Load a random or sequence container into the Property Editor.
2. In the Play Mode group box, select the **Step** option.

The container will only play one object each time it is played.

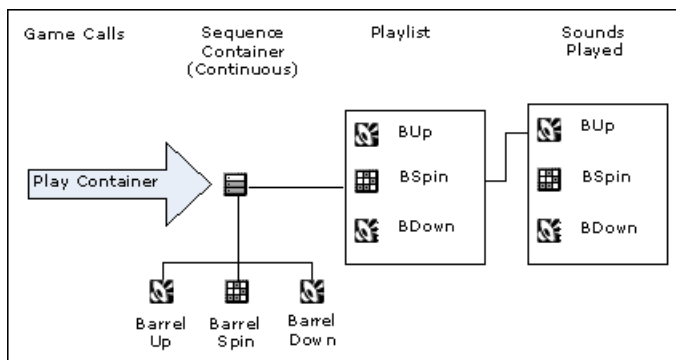
Related Topics

- [Playing All Objects Within the Container](#)
- [Defining the Playback Behavior for Random/Sequence Containers](#)
- [Creating a Random Container](#)
- [Creating a Sequence Container](#)
- [Creating a Playlist](#)
- [Defining the Scope of a Container](#)

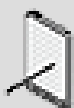
Playing All Objects Within the Container

In other situations, you will want all the objects within the container to be played each time it is called. For example, when certain guns within your game are fired, there are a series of sounds that must be played in sequence. In these situations, you would set the play mode of the container to Continuous.

The following illustration shows how Wwise manages the sequence of sounds played when a particular gun is fired using a sequence container in continuous mode.



In Continuous mode, all objects within the container are played, which means you can also add looping and various transitions between the objects.



Note

When a random container is in Continuous mode, some objects may be repeated several times before the complete list has played once. This is due to the weighting applied to each object within the container.

To play all objects within the container:

1. Load a random or sequence container into the Property Editor.
2. In the Play Mode group box, select the **Continuous** option.

The Continuous options become available.

3. For sequence containers, select the **Always reset playlist** option to return the playlist to the beginning each time the container is played. If you clear this option, the container will continue playback from where it was stopped or more precisely at the beginning of the next object in the playlist.
4. Select the **Loop** option to loop the entire contents of the container.

The Loop options become available.

5. Select one of the following options:

Infinite to specify that the container will be repeated indefinitely.

No. of Loops to specify a particular number of times that the container will be played.

6. If you selected the No. of Loops option, type the number of times you want the container to be played.
7. Select the **Transitions** option to apply a transition between the objects in the playlist.

The Transition options become available.

8. From the Type list, select one of the following options:

Xfade (amp) to add a crossfade between two objects using constant amplitude.



Xfade (power) to add a crossfade between two objects using constant power.



Delay to add a silence between two objects.

Sample Accurate to create a seamless transition with no latency between objects. Please note that the Vorbis and XMA audio formats are not reliable codecs for sample accurate transitions.

Trigger rate to define a specific rate at which the objects within the container will be triggered. This option is useful for simulating rapid gun fire.



Note

There are several limitations and restrictions when using crossfade, sample accurate, and trigger rate transitions. It is a good idea to generate an integrity report to see if these have affected your container.

9. In the Duration text box, type the length of time you want for the crossfade, delay, or trigger rate.



Note

The Duration option is not available for sample accurate transitions.

Related Topics

- [Playing One Object Within the Container](#)
- [Defining the Playback Behavior for Random/Sequence Containers](#)
- [Creating a Random Container](#)

- [Creating a Sequence Container](#)
- [Creating a Playlist](#)
- [Defining the Scope of a Container](#)

Defining the Scope of a Container

Since you may use the same container for several different game objects, you need to decide whether all instances of the container used in the game will be treated as one object or each instance is treated independently. In Wwise, this concept is called the scope of your container. You can set the scope to either of the following options:

- **Global** - Treats all instances of the container used in the game as one object so that repetition of sounds, voices, or motion FX objects across game objects is avoided.
- **Game object** - Treats each instance of the container as a separate entity, which means that no sharing of sounds or motion FX objects occurs across game objects.



Note

The Scope option is not available for sequence containers in Continuous play mode because the entire playlist is played each time an event triggers the container.

Defining the Scope of a Container - Example

Let's say you are creating a first person role-playing game. You have 10 guards that all share the same 30 pieces of dialogue. In this case, you can group your 30 sound voice objects into a random container that is set to Shuffle and Step. You would use this same container for all 10 guards and set the scope of the container to Global to avoid any chance that the different guards may repeat the same piece of dialogue. You can apply this concept to any container that is shared across objects in your game.

To set the scope of the container:

1. Load a random or sequence container into the Property Editor.
2. In the Scope group box, select one of the following options:

Global to treat all instances of the container used in the game as one object so that repetition of sounds, voices, or motion FX objects across game objects is avoided.

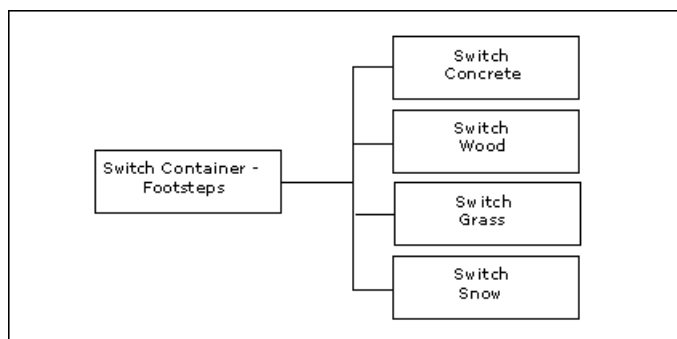
Game object to treat each instance of the container as a separate entity and, therefore, no sharing of sounds, voices, or motion FX objects occurs across game objects.

Related Topics

- [Defining the Playback Behavior for Random/Sequence Containers](#)
- [Creating a Random Container](#)
- [Creating a Sequence Container](#)
- [Creating a Playlist](#)
- [Defining How Objects Within a Container are Played](#)

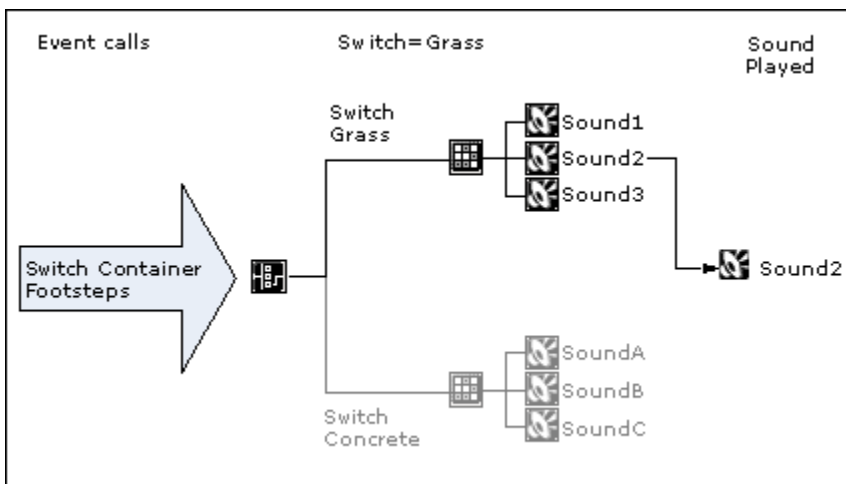
Defining the Contents and Behavior of Switch Containers

Switch containers allow you to group sounds or motion FX objects according to the different alternatives that exist within a game. Each alternative is represented in the switch container by a switch or state. For example, a switch container can be created for all the different surfaces that a character can walk on. The container might contain switches for concrete, wood, grass, snow, and any other surface that a character may come across in-game.



Within each switch/state are the audio or motion objects related to that particular alternative. For example, all the footstep sounds on concrete would be grouped into the “Concrete” switch, all the footstep sounds on wood would be grouped into the “Wood” switch, and so on. When the game calls the switch container, Wwise verifies which switch/state is currently active to determine which container or sound to play.

The following illustration demonstrates what happens when an event calls a switch container called “Footsteps”. This container has grouped the sounds according to the different surfaces a character can walk on in game. In this example, there are two switches: Grass and Concrete. When the event calls the switch container, the character is walking on grass (Switch=Grass), so the footstep sounds on grass are played. A random container is used to group the footstep sounds within the switch so that a different sound is played each time the character steps on the same surface.



Defining the Type of Switch Container

When creating a switch container, you must decide whether the container will be based on states, switches, or RTPCs. While you can select the switch or state options in the Switch Container Property Editor, RTPCs are linked to switch groups in the Switch Group Property Editor. For more information on how RTPCs can be associated with switches, refer to [Mapping Game Parameter Values to Switches](#).

After determining the switch type, you must also assign a switch or state group to the container. This defines the switches/states/RTPCs to which sounds or motion FX objects can be assigned.

Before you can assign a state or switch group to the container or use RTPCs for switches, you must create them first. For information on creating switch groups, state groups, and RTPCs, refer to:

- [Working with States](#)
- [Working with Switches](#)
- [Creating a Game Parameter](#)

To define the type of switch container:

1. Load a switch container into the Property Editor.
2. In the Switch Type group box, select one of the following options:

Switch to base the container on game switches.

State to base the container on game states.

3. From the Group list, select the switch or state group that you want to assign to the container.

The switches/states within that group appear in the Assigned Objects pane of the Contents Editor.

4. From the Default Switch/State list, select the switch/state that will be played when the game cannot identify a specific switch or state.

Related Topics

- [Defining the Contents and Behavior of Switch Containers](#)
- [Defining the Playback Behavior of the Switch Container](#)
- [Managing the Contents of a Switch/State](#)
- [Defining the Playback Behaviors of Objects Within a Switch Container](#)

Defining the Playback Behavior of the Switch Container

Since switches or states can change at any point in the game, you need to decide whether the sound or motion FX object will change immediately, or only the next time the switch container is played. The following play modes are available in Wwise:

- Step
- Continuous

You can use the Step option for one-shot sounds or motion, such as footstep sounds. The Continuous option, on the other hand, is more useful for objects that are continually looping, such as snowboarding sounds.

To define the play mode of a switch container:

1. Load a switch container into the Property Editor.
2. From the Play Mode group box, select one of the following options:

Step to play a new sound or motion FX object only after a new play event is triggered regardless of whether the switch changed during playback.

Continuous to play a new sound or motion FX object as soon as a new switch/state is detected. When Continuous is selected, a new play event is not required to change the object that is played.

Related Topics

- [Defining the Contents and Behavior of Switch Containers](#)
- [Defining the Type of Switch Container](#)
- [Managing the Contents of a Switch/State](#)
- [Defining the Playback Behaviors of Objects Within a Switch Container](#)

Managing the Contents of a Switch/State

You must assign the objects within the switch container to a particular switch or state. When assigning objects to switches/states, you may need to do any of the following:

- [Adding and Removing Objects from a Switch/State](#)
- [Adding and Removing Objects from a Switch/State](#)
- [Moving Objects Between Switches/States](#)

Adding and Removing Objects from a Switch/State

You can easily assign objects to switches by adding sounds, motion FX objects, and other containers to the **Assigned Objects** pane of the **Contents Editor**.

To help speed up the process, you can Ctrl or Shift+click several objects and then add or remove them all at the same time. If you assign several objects to a switch, they will all be played back simultaneously within Wwise and at run time in your game.

In most cases, you will assign the objects that are already in the switch container, but you may want to drag objects directly from the **Audio** tab in the **Project Explorer** to a switch/state. In this case, the objects are moved from their current location to the switch container. If you want to create a copy of the object instead of moving it, you can Ctrl+drag the object from the Audio tab in the Project Explorer to the switch/state in the Assigned Objects pane.

To assign/remove objects from a switch/state:

1. Load a switch container into the **Property Editor**.

The objects within the container are displayed in the **Contents Editor**.

2. To assign an object to a switch/state, drag an object from the **Contents** pane to a switch/state in the **Assigned Objects** pane.

The object is added to the switch/state.



Note

You must drop the object directly on the title of the switch or state.

3. To remove an object from a switch/state, click the object you want to remove in the **Assigned Objects** pane.
4. Press the **Delete** key.

The object is removed from the state/switch, but still remains in the **Contents** pane.

Related Topics

- [Moving Objects Between Switches/States](#)
- [Defining the Contents and Behavior of Switch Containers](#)
- [Defining the Type of Switch Container](#)
- [Defining the Playback Behavior of the Switch Container](#)
- [Defining the Playback Behaviors of Objects Within a Switch Container](#)

Moving Objects Between Switches/States

If you make a mistake when first assigning an object to a switch/state or just want to move objects, you can do so at any time.



Tip

To move several objects at the same time, Ctrl+click each object and then drag them to the new location.

To move objects between switches/states:

1. In the Assigned Objects pane of the **Contents Editor**, select the object or objects that you want to move.
2. Drag the object to the new switch/state.

The object is now assigned to the new switch/state.

Related Topics

- [Adding and Removing Objects from a Switch/State](#)
- [Defining the Contents and Behavior of Switch Containers](#)
- [Defining the Type of Switch Container](#)
- [Defining the Playback Behavior of the Switch Container](#)
- [Defining the Playback Behaviors of Objects Within a Switch Container](#)

Defining the Playback Behaviors of Objects Within a Switch Container

Since switches and states can change frequently within a game, you need to determine how each sound or motion FX object within the switch container will react when the change occurs. You can determine the playback behavior for the following options:

- **Play** - Determines whether an object will play each time the switch container is triggered or just when a change in switch/state occurs.

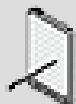
- **Across Switches** - Determines whether an object that is in more than one switch will continue to play when a new switch/state is triggered.
- **Fade In** - Determines whether there will be a fade in to the new object when a new switch/state is triggered.
- **Fade Out** - Determines whether there will be a fade out from the existing object when a new switch/state is triggered.

To define the playback behaviors of objects within a switch container:

1. Load a switch container into the Property Editor.

The objects within the container are displayed in the **Contents Editor**.

2. In the Play column, select the **1st only** option to play the object only the first time after the switch/state changes. If you leave the 1st only option unselected, the object will play each time the switch container is triggered by the game.
3. In the Across Switches column, select the **Continue to play** option to force the object that is in both the source and destination switches/states to continue playing during a change in switch/state. If you leave the Continue to play option unselected, the object will stop and will start playing again from the beginning.



Note

The Across Switches option is only available in Continuous play mode.

4. If you want to fade in to new objects when a change in switch/state occurs, type the amount of time you want in the Fade In text box.



Note

The Fade In option is only available in Continuous play mode.

5. If you want to fade out from existing objects when a change in switch/state occurs, type the amount of time you want in the **Fade Out** text box.

Related Topics

- [Defining the Contents and Behavior of Switch Containers](#)
- [Defining the Type of Switch Container](#)
- [Defining the Playback Behavior of the Switch Container](#)
- [Managing the Contents of a Switch/State](#)

Defining the Contents and Behavior of the Blend Container

You can use blend containers to group multiple objects so that they may be heard simultaneously. You can also group objects into blend tracks within a blend container so that you can easily apply one or more RTPC curves to them. You can even adjust the RTPC-based crossfading between these objects. This will allow you to create smooth transitions between sounds and motion FX objects as the parameters in your game change.

When working with blend containers, you can do the following:

- [Creating a Blend Container](#)
- [Working with Blend Tracks](#)
- [Managing Crossfades](#)
- [Auditioning the Contents of a Blend Container](#)

Creating a Blend Container

In certain game situations, you might want to have several related sounds or motion FX objects playing at the same time to create a complex composition. Blend containers are flexible structures you can use to group multiple objects. When the container is played, all the objects within it are played simultaneously.

You can add the following objects to blend containers:

- Sounds
- Motion FX objects
- Random containers
- Sequence containers
- Switch containers
- Blend containers.

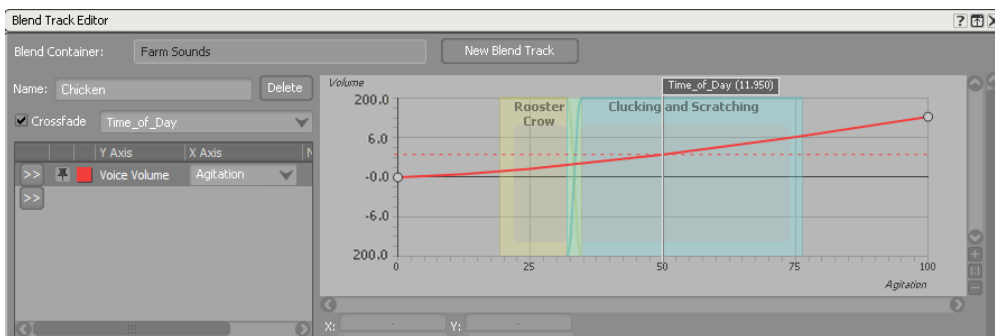
Using Blend Containers - Example 1

Let's say part of your game takes place on a farm. You want your players to feel like they are walking through a realistic farm setting. One way to encourage this mood would be to find a variety of farm animal sounds (cow moos, chicken clucks, horse whinnies) and group them into a "Farm Sound" blend container so they can be heard simultaneously.

Using Blend Containers - Example 2

Now, let's take your farm setting a step further. When your players walk through the farm gates, you want them to hear a range of sounds, not just a cacophony of animal noises. One way to solve this problem is to organize your sounds

within blend tracks and crossfade them based on a game parameter such as the time of day. For example, you can create a “Chicken Blend Track”, so your player hears roosters crowing in the morning, clucking and scratching sounds during the day, and silence at night. You can also create RTPCs to associate characteristics of your sounds to game parameters. For example, you might want your chickens to get nervous when you wave your sword around. To do this, you could create an “Agitation” RTPC, and create an RTPC curve that increases the volume of sounds in the Chicken Blend Track as the Agitation level increases.



Working with Blend Tracks

Blend tracks are provided for grouping objects and their corresponding property values within your blend container. Each blend track can contain many objects, which can then be heard all at once or as determined by game parameters using RTPCs.

When you are working with blend tracks, you will need to do the following:

- [Creating Blend Tracks](#)
- [Adding and Removing Objects from Blend Tracks](#)

Creating Blend Tracks

The first step in organizing objects within your blend container is to create and name individual blend tracks.

To create a new blend track:

1. Load a blend container into the Property Editor.
2. Click **Edit Blend Tracks**.

The Blend Track Editor opens.

3. Click **New Blend Track**.
4. Type a name for the blend track and press **Enter**.

The new blend track is displayed in the Blend Track Editor.

5. Repeat this procedure to create new blend tracks as needed.

The blend tracks you create are displayed in the **Blend Track Editor**, and in the **Blend Tracks** section of the **Contents Editor**.



Note

To delete a blend track, select that blend track and press **Delete**.

Related Topics

- [Adding and Removing Objects from Blend Tracks](#)
- [Adding RTPC Curves to Blend Tracks](#)
- [Editing RTPC Curves in a Blend Track](#)
- [Displaying RTPC Curves in a Blend Track](#)

Adding and Removing Objects from Blend Tracks

The blend tracks you create remain empty until you populate them with objects. Each blend track in a blend container can contain up to 128 objects. One object can exist in multiple blend tracks at the same time, and one blend track may contain the same object multiple times. You can add and remove objects to and from blend tracks in the **Contents Editor**.



Tip

The order of objects in a blend track is important because it determines how they will be positioned for crossfading. For more information on crossfading, refer to [Managing Crossfades](#).

To add objects to blend tracks in a blend container:

1. Load a blend container into the **Property Editor**.
2. Populate each blend track by dragging objects from the **Contents Editor** or the **Project Explorer** to each blend in the **Blend Tracks** list.



Note

To remove an object from a blend track, select the object and press **Delete**. This removes the object from the **Blend Track** list but does not remove the object from the **Contents Editor**.

Related Topics

- [Adding RTPC Curves to Blend Tracks](#)
- [Editing RTPC Curves in a Blend Track](#)
- [Displaying RTPC Curves in a Blend Track](#)

Adding RTPC Curves to Blend Tracks

There are two ways to add RTPC curves to blend tracks:

- Adding curves to the blend container itself.
- Adding curves to each blend track within a blend container.

If you choose to add an RTPC to a blend container itself, that RTPC will be applied equally to all objects within the container. For more information on creating and working with RTPCs, refer to [Overview](#).

However, you can also add RTPCs to each blend track in a container. In this way, you can apply RTPCs to selected objects at a time. For example, say you've added all the collision sounds in a racing game to one blend track. If you add an RTPC to this blend track that varies volume according to impact force, the volume of each collision sound will be affected by this RTPC.

To add an RTPC curve to a blend track:

1. In the Blend Container Property Editor, click **Edit Blend Tracks** to open the Blend Track Editor.



Tip

You can also press Ctrl+Shift+T to access the Blend Track Editor.

The blend tracks you have previously created are displayed.

2. In the blend track to which you want to add RTPC curves, click the **Property Selector** button.

A list of available sound properties is displayed.

3. Click the sound property you want to be affected by the RTPC.

An RTPC curve is displayed in the graph view.

4. From the X axis list, click the game parameter that you want to assign to the Wwise property.

The game parameter is assigned to the X axis in the graph view.

5. Repeat these steps to add more RTPC curves to the graph.



Note

As you add multiple curves with different game parameter units, the units will disappear from the axes. However, you will still be able to see the units for any selected point displayed in the X axis and Y axis boxes below the graph.

Related Topics

- [Adding and Removing Objects from Blend Tracks](#)
- [Editing RTPC Curves in a Blend Track](#)
- [Displaying RTPC Curves in a Blend Track](#)

Editing RTPC Curves in a Blend Track

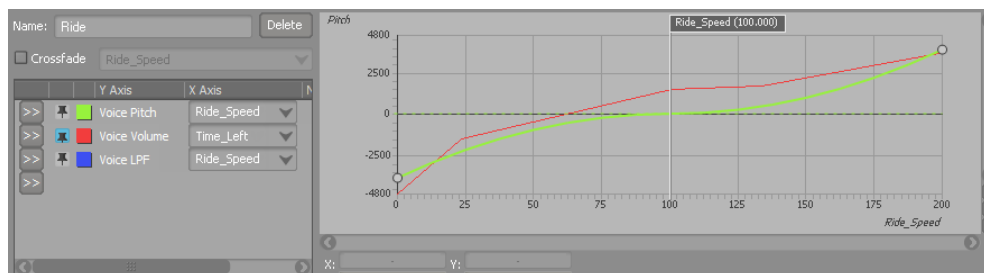
After you have added your RTPC curve or curves to a blend track, you can define them as you would any other RTPC curves. For example, you can add points, move them, or define curve segments. For information on how to do this, refer to [Controlling Property Values Using Game Parameters](#).

Related Topics

- [Adding and Removing Objects from Blend Tracks](#)
- [Adding RTPC Curves to Blend Tracks](#)
- [Displaying RTPC Curves in a Blend Track](#)

Displaying RTPC Curves in a Blend Track

Each blend track of a blend container may contain multiple RTPC curves. Using the Blend Track Editor, you can work with multiple RTPC curves at once.



When you select an RTPC curve in RTPC list, it is highlighted in the graph view and can be edited. For information on working with RTPC curves, refer to [Controlling Property Values Using Game Parameters](#).

Related Topics

- [Adding and Removing Objects from Blend Tracks](#)
- [Adding RTPC Curves to Blend Tracks](#)
- [Editing RTPC Curves in a Blend Track](#)

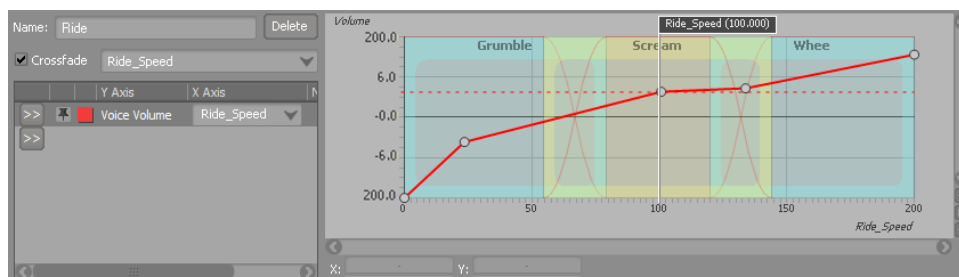
Managing Crossfades

Blend containers allow multiple sounds or motion FX objects to be played at once. Enabling crossfading in a blend track changes how these objects are heard or felt. To help you manage crossfading, each object in a blend track is displayed as a block. These blocks are charted on a graph with an X axis representing a changing game parameter. The position of these blocks on the X axis indicates the game parameter values at which the sounds will be heard or the motion FX objects will be felt.

When you overlap blocks to create crossfades, you smooth the transition between objects and increase realism in your game. Crossfading can also be combined with RTPC curves in the blend tracks of blend containers to alter the properties of objects.

Using Crossfades - Example

For example, let's say your game is a roller-coaster simulator. Your player can build coasters with speeds that vary from 0 to 200 km/h. On a basic level, you can use a blend container to assign an RTPC to the sounds of the roller coaster riders, increasing their volume as the coaster speed increases. But let's say you've collected many different sounds to represent the riders, ranging from bored grumbling to excited cheering to terrified screams. If you use a blend container to hold these sounds, these different sounds can be heard selectively depending on what speed the coaster is going. When the coaster speed increases to 40 km/h, for example, your excited cheering sounds could begin to be heard. Your player will hear the different rider sounds, as well as an increase in volume for each sound.

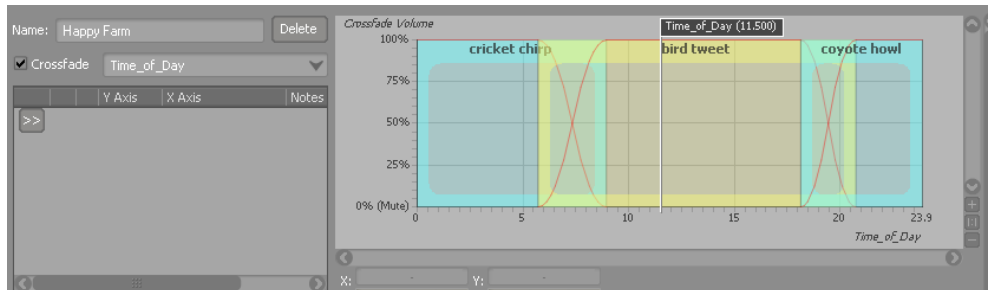


Using Crossfading

To use crossfades in a blend container, you first need to enable them. Then, you can adjust the size of your blocks. When the game parameter reaches the point

where the block meets the X axis, the object is heard. When you overlap these blocks, you create a crossfade.

Crossfades help organize the playback of sounds and motion FX objects within a blend container. Because all the objects in a blend container play at once and are heard or felt based on RTPC mappings, using crossfades creates room for smoother transitions between objects. For example, if you created an RTPC to represent the passage of time in your game, you could overlap crossfades for cricket sounds and bird sounds to represent dawn.



To use crossfading:

1. In the Blend Track Editor, select the blend track to which you want to add crossfading.

The blend track becomes active.

2. Select the Crossfading option.

The Game Parameter list becomes available.

3. Select the game parameter on which you want to base crossfading in this blend track.

Blocks are displayed for the objects in the blend track, with the X axis representing the game parameter.

Each object in the blend track is represented as a colored block in the graph. The blocks are drawn in the same order as the objects are listed in the blend tracks column of the contents editor.

4. Adjust block sizes and positions by selecting a block and dragging its edges left or right.

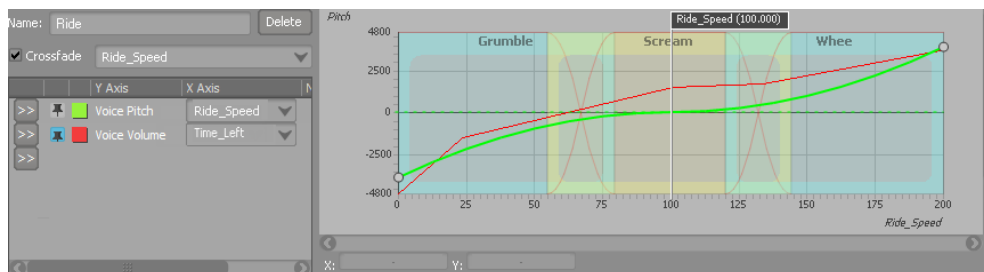
The objects corresponding to the blocks are played as indicated by the X axis.

Related Topics

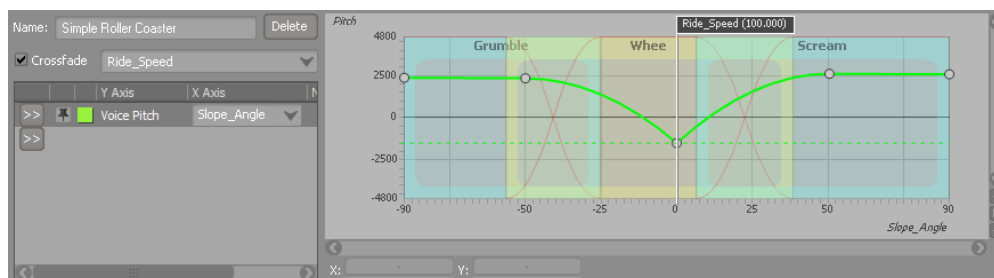
- [Managing Crossfades](#)
- [Using Crossfading with RTPCs](#)
- [Setting Crossfade Modes](#)

Using Crossfading with RTPCs

Blend tracks allow you to combine the power of RTPC curves with the flexibility of crossfades. Within a blend container, you can determine the crossfading between objects based on a game parameter, and then apply RTPC curves to these same objects.



Keep in mind that you can base crossfades and RTPC curves in a blend track on different parameters. For example, in a roller-coaster simulator, you could have an RTPC curve that increases the pitch of your rider noises as the car angle becomes steeper. However, you could add cross-fading to that blend track based on speed to change which sounds are heard based on speed. Therefore, the sounds that are heard shift as the ride gets faster, and all these sounds become higher pitched as the ride gets steeper.



Related Topics

- [Using Crossfading](#)
- [Managing Crossfades](#)
- [Setting Crossfade Modes](#)
- [Using Caution with Crossfade Transitions](#)

Setting Crossfade Modes

Wwise can handle crossfading in various ways, all depending on your preferences. You can use the following three modes to adjust the crossfading between the objects in your blend tracks:

- **None:** The property of the RTPC transitions from its maximum to its minimum value or vice-versa instantly when crossing the overlapping block's boundaries.

- **Automatic (default):** The fading range depends on the width of the overlap with the neighboring block. If there is no neighboring block, there is no fade.
- **Manual:** You can move the point where the fade-in ends or the fade-out begins. The beginning of a fade-in or end of a fade-out will always begin at the outside lower corner of the overlapping block.



Note

You can also modify the shape of the crossfade curves. For information on setting curve shapes, refer to [Specifying the Shape of the Curve Between Control Points](#).

To select a crossfade mode:

1. Right-click the edge of a block and click a mode.

Related Topics

- [Using Crossfading](#)
- [Managing Crossfades](#)
- [Using Crossfading with RTPCs](#)
- [Using Caution with Crossfade Transitions](#)

Using Caution with Crossfade Transitions

The following listed points provide some additional information that you should be aware of before using the crossfade transition within a **Random** or **Sequence Container**.

- Audio File Length
 - Audio file length must be greater than or equal to 0.2 seconds.
 - The minimum crossfade time is 0.1 seconds.
- Crossfade time in relation to audio file length
 - When cross-fading from Sound A to Sound B, the maximum crossfade time allowed by the sound engine is half the length of audio file A. In cases where the crossfade time is longer than the maximum allowed, the crossfade time will automatically be adjusted to half the length of the outgoing audio file.



Note

Wwise does not limit or indicate that the crossfade is too long for one or more audio files in the container. If an

adjustment to the crossfade time is required, it will be done by the sound engine at runtime.

- Pitch and Crossfades
 - If you use an RTPC to set the pitch value for a container or a Set Pitch event action is triggered while a container is playing, you may experience unexpected results when the crossfade is applied between sounds.
- Source Plug-ins and crossfades
 - If you apply a crossfade to a source plug-in, the crossfade may be ignored if the end of the source cannot be determined. This can happen, for example, when the duration of a sine generated source is based on an RTPC. In these types of cases, the crossfade is ignored and the transition is done without the crossfade.
- Switch Containers and Crossfades
 - When a **Switch Container** is a child of a **Sequence Container**, crossfade transitions will be applied differently depending on the number of Wwise objects assigned to a switch.
- Two Voices
 - Two different voices are used by the sound engine during a crossfade.
- Virtual Voices and Crossfades

In sum, you should avoid using these types of virtual voices with containers that use crossfade transitions. If you wish to use virtual voices with these containers, you should select the **From Elapsed Time** behavior.

- By definition, **From Beginning** and **Resume** virtual voice behaviors have an effect on sound duration when it goes under the volume threshold, which is not taken into account by the crossfade timing mechanism.
- Voices may become virtual when their volume goes under the threshold. For any given sound, the volume that is compared against the threshold is the real effective volume of all its audio channels, resulting from the contribution of all volumes of the **Actor-Mixer Hierarchy**, fade transitions, interactive-music transitions, RTPCs, States, positioning, and attenuations.
- The contribution of fade transitions also counts when computing the effective volume of a sound. Consequently, during a crossfade transition inside a **Random**, **Sequence**, or **Blend Container**, fading sounds will likely go below the volume threshold for some time. If their behavior under threshold is "**From Beginning**" or "**Resume**", their real duration will be longer than expected by the container's logic. This will result in unpredictable behavior. Worse, when a sound fades out under threshold, it stops "virtually" playing, and therefore never ends. The container could, consequently, stop sequencing more sounds.

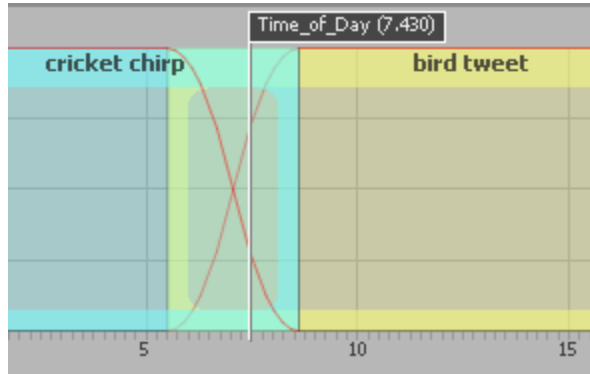
Related Topics

- [Using Crossfading](#)
- [Managing Crossfades](#)
- [Setting Crossfade Modes](#)
- [Using Crossfading with RTPCs](#)

Auditioning the Contents of a Blend Container

You can use blend containers to create complex and realistic soundscapes, so you will probably want to test them as you go along. By using the Transport Control and the Blend Track Editor, you can audition the contents of a blend container. You can then make adjustments on the fly, such as resizing crossfades or adding RTPC curve points.

In the Blend Track Editor, a cursor is provided so that you can change the game parameters while you are playing back the blend container. Since you have already mapped these objects to Wwise property values, when you change the game parameter values, you automatically change the object property values. This simulates what happens in game when the game parameters change so you can verify how effectively your property mappings will work in game.



Note

When you play back a blend container, all the objects in it are played back, whether or not they are associated to a blend track.

To play the contents of a blend container:

1. Load the blend container into the Transport Control.
2. In the Blend Container Property Editor, click **Edit** to open the Blend Track Editor.



Tip

You can also press Ctrl+Shift+T to open the Blend Track Editor.

The blends of the blend container are displayed.

3. Click the **Play** icon.

While the blend container is playing, you can use the cursor to change the game parameter values to hear or feel how your sounds or motion FX objects react to the changes.



Tip

While tweaking the RTPC curves of the individual objects within your blend container, you can continue to audition the blend container by pinning it in the Transport Control. For information on pinning, refer to [Pinning an Object in the Transport Control](#).

Related Topics

- [Creating a Blend Container](#)
- [Working with Blend Tracks](#)
- [Adding and Removing Objects from Blend Tracks](#)

Object Playback Tips & Best Practices

Before deciding on the particular playback behaviors for the objects in your project hierarchy, you may want to review the following sections, which provide you with a series of tips and best practices that can help you get the results you are looking for.

Streaming Sounds

As a general rule, you will want to stream sounds that are too large to store in memory, for example music, ambient room tones, and even voices, if there are many of them. However, many different things come into play when determining which option is best.

At the beginning of your project, you should speak to the development team to determine the following:

- How many streams will be available for audio?

- What is the bandwidth of each stream?
- Will these streams be shared?
- How many concurrent streams can you use?
- Will the streaming occur from the platform hard drive or directly from the DVD?

With this information, along with a sound's sample rate and compression format, you can perform some calculations and then make informed decisions about whether to stream sounds or not.

Streaming and Prefetch

The more streams you have playing concurrently, the more pre-fetch time will be required. You can start out using a short pre-fetch time, but as the number of concurrent streams increases over the course of your project, you might need to increase it.

Random Containers

Avoiding repetition of sounds within random containers that have a scope set to “global” - the purpose of using a continuous random container in shuffle mode is to avoid the repetition of sounds. If, however, your container has a limited number of sounds and you have many instances of this container playing back, you may experience some repetition. To avoid this type of behavior, make sure that the number of sound objects within the container is at least twice the number of playback instances. For example, if you have 3 playback instances of the same random container, your container should contain at least 6 different sound objects to avoid repetition.

Blend Containers

- CPU usage - be aware that when you play a blend container, all the objects in it will play simultaneously and can therefore use up a large amount of memory. You can use the virtual voices settings to reduce CPU use, but this may result in glitches if crossfades are short and frequent. For more information on using virtual voices, refer to [Volume Threshold and Virtual Voices](#).
- Deciding between switches and blend containers - during gameplay, both switch containers and blend containers can be used with RTPCs and produce similar results. However, there are significant differences in how the two features function, and you should choose between them depending on your requirements.
 - If you want different sounds or motion FX objects to be played at different times, and crossfades are unimportant, you can use a switch container.
 - If you want all sounds or motion FX objects to be played at all times, and if crossfades are required, use a blend container.

Chapter 10. Defining Positioning for Sound and Motion

Overview	262
Understanding Positioning in Wwise	263
Working with 2D Sound, Music, and Motion FX Objects	267
Working with 3D Sound, Music, and Motion FX Objects	269
Applying Distance-Based Attenuation	271
Defining Spatial Positioning Using Animation Paths	287
Routing Audio Signals to the Center Speaker	298
Positioning Tips and Best Practices	299
Understanding Channel Configurations	306
Speakers vs Headphones Panning Rules	308

Overview

The positioning and propagation of sounds, music, and motion play a key role in engaging players and immersing them in your game; therefore, it is important to understand how to deal with the many types of sounds and motion effects that you will have in your game.

A typical game will have a combination of the following types of sounds and motion effects:

- **Localized ambient sounds and motion** - Where the sound or motion emitter remains in one location. For example, localized ambient sounds can include a large machine or fountain.
- **Non-localized ambient sounds and motion** - Where the sound or motion emitter moves, but is not attached to a particular game object. For example, non-localized ambient sounds can include ambient bird or insect sounds.
- **Mobile object sounds and motion** - Where the sound or motion emitter moves with a particular game object. For example, mobile object sounds can include any sound triggered by a game character, animal, and so on, such as a barking dog or a shouting guard.
- **Game interface sounds and motion** - Where the sound or motion is associated with a particular game interface element or other item that maintains a fixed position on the screen. For example, game interface sounds and motion effects can include parts of a Heads-Up Display (HUD), menu sounds (buttons, navigation), or the gun in a first-person shooter game.

Wwise has a powerful and flexible toolset for positioning that will allow you to deal with each of these sound and motion types in a way that will create the experience gamers are expecting with next-generation games.

Positioning - Example (Part 1)

Let's say you are creating a first-person stealth game. At one point in the game, a group of special agents must travel to a remote volcanic island where terrorists are holding one of your agency's operatives. This mission is very dangerous so the agents must work as a team and stay close together. As the agents sneak through the enemy's jungle base, they experience the following sounds and motion:

- The main character's footsteps.
- The torches that light up the enemy's jungle base.
- A group of terrorists talking in a hut.
- A mosquito buzzing overhead.
- The updates received from headquarters.
- The whispered communication between special agents on this mission.

- The detonation of explosives used to destroy the base after the mission has been successfully completed.
- The constant rumbling of the island's volcano.
- The interactive music.

Each of these sounds and motion effects will require a different treatment in terms of positioning and propagation. The following sections discuss how the positioning for each of these types of sounds and motion effects can be managed in Wwise.

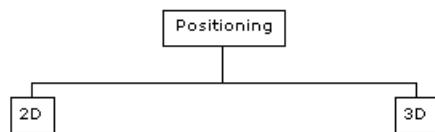
Understanding Positioning in Wwise

Wwise has a variety of tools that can help you simulate realistic positioning and propagation for any sound, music, or motion FX object in your game. The following sections describe how positioning works for the various objects in Wwise:

- [Positioning Sound and Music Objects](#)
- [Creating Multiple Positions for a Single Game Object](#)
- [Positioning Motion FX Objects](#)
- [Setting Positioning Options within the Project Hierarchy](#)
- [Handling multichannel sources with 3D positioning](#)

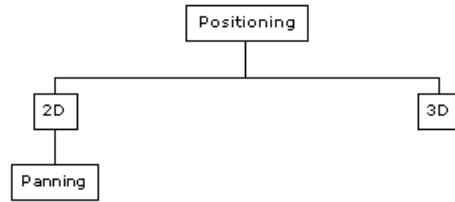
Positioning Sound and Music Objects

At a very basic level, your sound and music objects can use either 2D or 3D positioning.



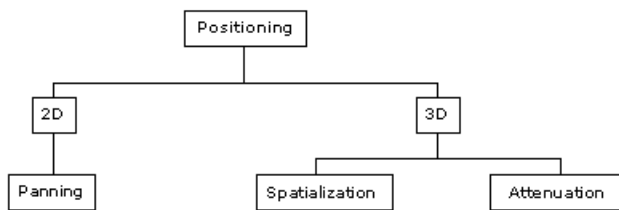
The main difference between the two types of positioning is the way in which the source channels are mapped to the output speakers. In 2D positioning, the source channels are linked together, whereas in 3D positioning, each input channel can be output to any speaker in a surround environment. The positioning method can be changed at runtime through an RTPC on the Positioning Type.

For 2D sound and music objects, you can use a panner to balance the volume of each channel so the sound or music object can be heard through different speakers.



For 3D sound and music objects, however, there are two separate things to consider when positioning your sounds: the actual position or location of the sound emitter within the 3D space and its distance from the listener. To deal with both of these issues, you have the following tools in Wwise:

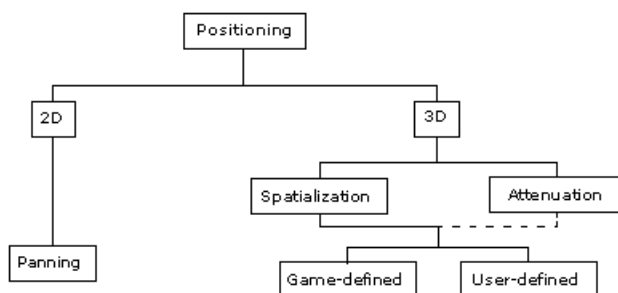
- **Attenuation** - Simulates the natural weakening of an audio signal as the sound emitter moves and/or turns away from the listener.
- **Spatialization** - Determines the actual location or positioning of the sound or music object within the 3D environment of the game.



For added flexibility, the spatialization information can be either of the following:

- **User-defined** - Where the positioning information is pre-defined in Wwise using specific animation paths.
- **Game-defined** - Where the positioning information is calculated using the actual position of the sound emitter in game.

The following illustration summarizes the different positioning options that are available for the sounds and music in Wwise.



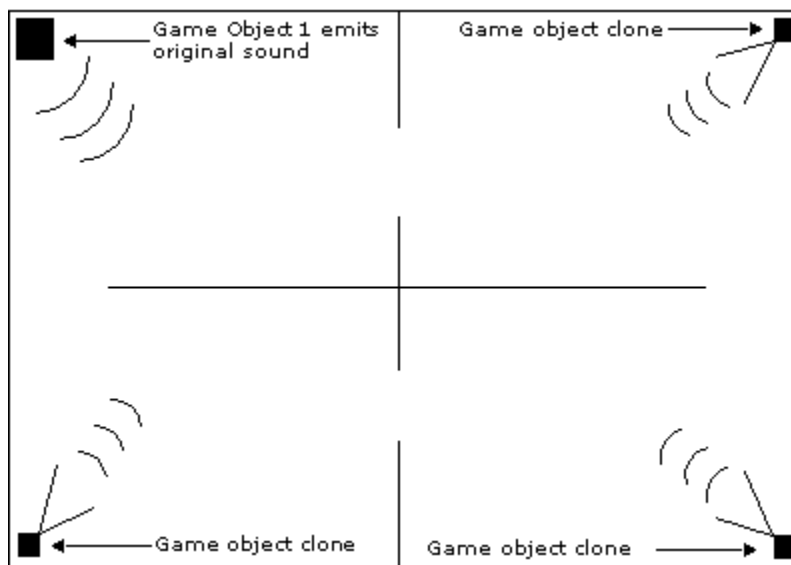


Note

Although the Game-defined and User-defined positioning information is used to determine the spatialization within the surround environment, it is also used to calculate the distance for attenuation.

Creating Multiple Positions for a Single Game Object

In certain situations in your game, a single instance of a sound may not be sufficient to create a realistic soundscape, for example, a PA system that has multiple speakers. If you don't want to duplicate the sound and corresponding game object, you can create a “clone” of the game object that emits the sound and position it at a different location in the game. In the PA system example, you can “clone” the game object that emits the original sound and reposition each clone to the location of the PA system's speakers, as shown in the following illustration.



Each clone produces an exact replica of the original sound and has the same properties, behaviors, and attenuation settings as the original. Since you have the original sound and the “clones” playing at the same time, you can either add their volumes together or specify that the cumulative volume of all sounds will be played at the maximum volume level of the original sound.

When using “clones”, you need to be careful when defining the spread and attenuation settings for your original sound as they will most likely be different than if you only had the sound emanating from a single position. This is especially true if the game player will be able to hear the original sound and the

“clones” at the same time. If the “clone” sounds are added to the original, the volume of the combined sounds may become too loud and result in clipping.

Other examples of when you might want to “clone” game objects are as follows:

- When you have many torches lining the walls of a corridor.
- When you have a large odd-shaped object, where simple radius attenuation is not suitable, such as a lake or water fall.
- When parts of a wall crumble exposing sounds being emitted from behind the wall. By positioning “cloned” game objects in specific locations, you can effectively simulate partial occlusion and obstruction.

Game objects can only be “cloned” and re-positioned using the Wwise SDK. For more information on setting multiple positions for a single game object, refer to the section [Sound Engine Integration Walkthrough > Integrate Wwise Elements into Your Game > Integrating 3D Positions > Integration Details - 3D Positions](#) in the Wwise SDK documentation.

Positioning Motion FX Objects

Positioning for less sophisticated devices, such as game controllers, is not an issue because the motors within these devices cannot simulate a 3D environment. You can, however, use attenuation to reduce the intensity of the motion signal as it moves away from the game player.

Setting Positioning Options within the Project Hierarchy

When setting the positioning for sound, music, or motion FX objects, keep in mind that each object within the hierarchy can have a positioning setting. Positioning is considered an absolute property, which means that the positioning of the top-level object is automatically passed down to all child objects. You can, however, override these settings if you need to customize the positioning for a particular container, sound, or motion FX object. For more information on how positioning works within the hierarchy, refer to [About Properties in the Project Hierarchy](#).



Note

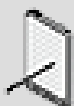
The Wii platform physically outputs a two-channel audio stream, however, it uses Dolby Pro Logic® II to simulate a five-channel surround sound experience. After DPL-II mode is enabled by the programmer at initialization time (by passing `AX_MODE_DPL2` as `AkPlatformInitSettings::uAXMode`), Wwise will use 4.0 speaker values for 3D positioned sounds. 4.0 speaker values are used because DPL-II does not allow you to directly address the center speaker. The positioning on the Wii platform, therefore, is restricted to the limitations of this technology. For more

information on these limitations, refer to DPL-II manual or white papers.

Handling multichannel sources with 3D positioning

Multichannel sounds can be positioned in space using the 3D positioning tab just as well as mono sources, without downmixing to mono.

In order to compute a source's channel contributions to each speaker, "virtual emitters" are set in a semicircle around the *listener*. The extent of that semicircle depends on the spread (100% is the whole circle, 0% is a point in front of the listener). This semicircle then is split in as many regions of equal size as there are channels, and each of the source's original channels contribute to their assigned region. For example, with a stereo source, all virtual emitters on the left of the circle use the source's left channel, and all those on the right use the source's right channel. Once this is done, the actual contribution of each virtual emitter is computed for each speaker. Thus, the multichannel image is somewhat preserved even with 3D positioning.



Note

In a case where the spread is 0, all virtual emitters are located in a single point in front of the listener, which is similar to downmixing all channels to mono before positioning it in the 3D space.

Working with 2D Sound, Music, and Motion FX Objects

By default, channels of 2D sounds and music are assigned directly to the corresponding channels of their output bus, regardless of the position or orientation of the listener or game object. You can, however, use the 2D Panner to balance the volume of each channel within the sound or music object so that they can be heard in varying levels through the channels of an output bus.

The 2D Panner contains a two dimensional graph view with x and y coordinates that simulates the "pan" and "fade" controls of a car panner. You can drag the black circle that represents the point source anywhere within this graph view to change the amount of power routed to the channels of the output bus.

The actual power distribution depends on the output bus configuration. For example, the y coordinate has no effect when the output bus is stereo. Note that side speakers of a 7.1 configuration are at their maximum level when the black circle is in the middle.

Some sounds, such as character voices, are key to the game play, so you may want to route their signals to the center speaker to ensure audibility. The Center

% controls allow you to define the amount of the signal that will pass through the center speaker. For more information on using the Center % controls, refer to [Routing Audio Signals to the Center Speaker](#).



Note

The Center % property has no impact on motion FX objects.

2D sounds and music can be used for a variety of purposes in your game, for example, menu sounds and character voices. They also use less CPU and memory, so you may want to consider using 2D sounds in situations where memory or CPU usage are issues.

2D Positioning for Motion FX Objects

For less sophisticated motion devices, such as game controllers, you should use 2D positioning without panning because the motors within these devices cannot simulate a 3D environment. There may be some situations, however, where you might want to reduce the intensity of a motion signal based on the proximity of the source. In these cases, you can set the positioning of your motion FX object to 3D and then use the attenuation settings.

To define the channels of a 2D sound, music, or motion FX object:

1. Load a top-level object into the Property Editor.
2. Switch to the Positioning tab.



Note

If the object is not a top-level object, you must select the Override parent option before you can set the Positioning options.

3. Select the 2D option.

The 2D positioning controls become available.

4. If you want to pan the signal to a speaker or motor other than the default front left and front right, select the Enable Panner check box.
5. Click **Edit** to open the 2D Position Editor.
6. Drag the black circle, which represents the point source, to any position within the 2D plane.

The volume or intensity of each channel will be adjusted to simulate the position you specified.



Note

You can specify exact values using the X and Y coordinate text boxes.

Related Topics

- [Working with 3D Sound, Music, and Motion FX Objects](#)
- [Routing Audio Signals to the Center Speaker](#)

Working with 3D Sound, Music, and Motion FX Objects

For 3D sound and music objects, each input channel can be output to any speaker in a surround environment making it easy to simulate movement of the object in relation to the listener. To achieve this enhanced surround experience, Wwise uses both spatial positioning and attenuation.

Spatialization determines the actual location or positioning of the object within the 3D environment of the game. When you use spatialization, movement of the source in game is reflected by hearing the sound through different speakers in a surround environment. If you choose not to use spatialization, a sound or music object is played according to its original channel configuration regardless of any movement of the source.

Depending on the type of sound you are dealing with, you can choose between two types of spatialization:

- **User-defined** - Where the positioning information is pre-defined in Wwise using specific animation paths.
- **Game-defined** - Where the positioning information is defined by the actual position of the object in game.

Most of the mobile sound and music objects in your game will use game-defined spatialization, but user-defined spatialization can be used for a variety of purposes too, for example, non-localized ambient sounds, such as birds or insects.

In addition to spatialization, you can define the attenuation of a 3D sound or music object. The attenuation settings simulate the natural weakening of a signal as it moves away from the listener. Wwise uses a series of curves to map Wwise property values, such as volume and low pass filter, to specific distance values. With these curves, you can create a sophisticated distance-based roll-off for your sound and music objects. To add even more realism, you can also use sound cones that attenuate the sound based on the orientation of the object in relation to the listener. For more information on defining the attenuation of your 3D objects, refer to [Applying Distance-Based Attenuation](#).

You can use a combination of the different positioning options to create a rich and diverse audio environment for your game. In certain situations, you may want to use only spatialization, in others you may want to use only attenuation, and in others you may want to use both.



Note

For the 3D sounds that are key to the game play, you may want to route their signals to the center speaker to ensure audibility. The Center % controls allow you to define the amount of the signal that will pass through the center speaker. For more information on using the Center % controls, refer to [Routing Audio Signals to the Center Speaker](#).

3D Positioning for Motion FX Objects

Not all motion devices fully support 3D positioning. Game controllers, for example, have few motors with a limited number of movements, and therefore, cannot make use of the 3D spatialization information. As a result, for less sophisticated motion devices, you should use 2D positioning without panning. There may be some situations, however, where you might want to reduce the intensity of a motion signal based on the proximity of the source. In these cases, you can set the positioning of your motion FX object to 3D and then use the attenuation settings without spatialization.

To define the properties of a 3D sound, music, or motion object:

1. Load a top-level parent object into the Property Editor.
2. Switch to the Positioning tab.



Note

If the object is not a top-level object, you must select the Override parent option before you can set the Positioning options.

3. Select the 3D option.

The 3D positioning controls become available.

4. In the Position Source group box, select one of the following options to determine the source of the spatialization information:

User-defined to pre-define the spatial positioning information in Wwise using animation paths.

Game-defined to have the spatial positioning information calculated in real time by the game.



Note

If you selected User-defined, click the **Edit** button to open the 3D Position Editor where you can define the animation paths for your sound, music, or motion FX objects. For more information on creating animation paths, refer to [Defining Spatial Positioning Using Animation Paths](#).

5. If you set the Position Source to User-defined, select the **Follow Listener Orientation** check box to lock the position of the animation path to the orientation of the listener. When this option is selected, the sound will always be heard through the same speakers regardless of the orientation of the listener. When this option is not selected, the listener moves independently of the path. This means that the sound will be heard through different speakers as the listener turns around. For more information on this option, refer to [Creating Animation Paths that Follow the Orientation of the Listener](#).
6. If you set the Position Source to Game-defined, select the **Update at each frame** check box to update the source's position information at every game frame. If you disable this option, the source's position remains static until it finishes playing.
7. If you want the spatial positioning information to be calculated to simulate movement within the 3D game space, select the **Enable Spatialization** check box. If you disable this option, a sound, music, or motion FX object is played according to its original channel configuration regardless of any movement of the source.

Related Topics

- [Applying Distance-Based Attenuation](#)
- [Defining Spatial Positioning Using Animation Paths](#)
- [Routing Audio Signals to the Center Speaker](#)

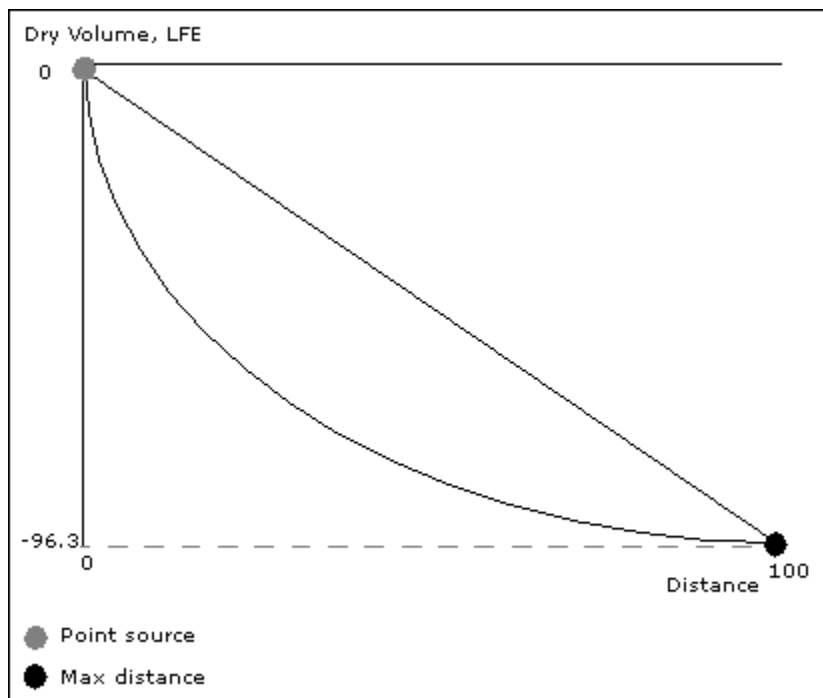
Applying Distance-Based Attenuation

To simulate a natural roll-off of audio or motion as the source moves away from the microphone in game, you can use the attenuation properties. The attenuation in Wwise is based on the following two properties:

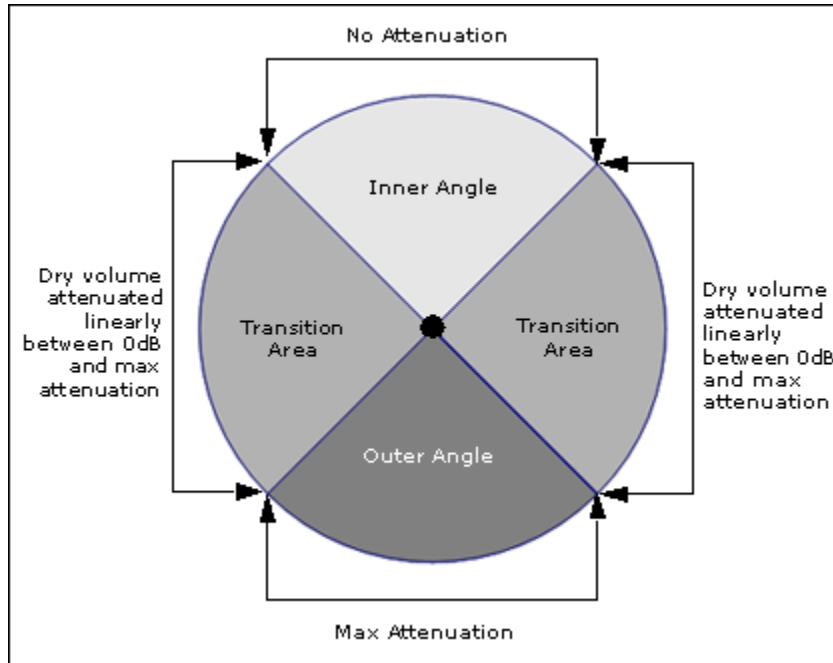
- **Distance attenuation** - Which influences the intensity of the signal based on the distance between the source and the listener.

- **Cone attenuation** - Which influences the intensity of the signal based on the orientation of the game object in relation to the listener.

The distance attenuation is defined using a series of curves. These curves map Wwise property values, such as volume, to a distance value. By defining the properties of each point along the curve, you can control the volume attenuation for sounds or motion objects as they move away from the listener.



The cone attenuation is defined using a series of angles that define areas in front of, beside, and in back of the source. By defining these different areas around the source, you can simulate attenuation of a sound or motion object based on the orientation of the object in game.



Like effects, you can create instances of your attenuation properties that can then be shared across many different objects within Wwise using ShareSets.

Managing Attenuation Instances

Attenuation instances are a collection of attenuation-related properties. Since many of the sound, music, and motion objects within your game will have the same attenuation properties, you can create an attenuation instance and then share it across many objects in your project using an attenuation ShareSet.

A ShareSet is a collection of instance properties to which objects subscribe. When changes are made to the ShareSet, all objects subscribing to that ShareSet are affected. The advantages of using a ShareSet are that you don't have to modify the attenuation properties for each object individually and you can save valuable memory in game.

In some situations, however, you may not want to share the attenuation settings. In these cases, you can create a custom attenuation instance that has unique property values.

You can carry out the following tasks to help manage the attenuation ShareSets in your project:

- [Creating an Attenuation ShareSet](#)
- [Deleting an Attenuation ShareSet](#)

Creating an Attenuation ShareSet

Before you can define the attenuation properties for an object, you must create an attenuation ShareSet. An attenuation ShareSet is a collection of attenuation property settings. ShareSets can be used by all objects that require the same kind of attenuation properties.

To create an attenuation ShareSet in the Project Explorer:

1. In the Project Explorer, switch to the **ShareSets** tab.
2. Under the **Attenuations** section, do one of the following:

Select a work unit or virtual folder and click the **Attenuation** icon in the Project Explorer toolbar.

Right-click a work unit or virtual folder and select **New child > Attenuation** from the shortcut menu.

A new ShareSet appears as a child of the work unit or virtual folder in the hierarchy.

3. Type a name for your new ShareSet and press **Enter**.

The new ShareSet is displayed in the Attenuations hierarchy.



Note

Each Attenuation ShareSet name must be unique. You can rename a ShareSet at any time by right-clicking it, selecting **Rename**, and typing a new name.

To create an attenuation ShareSet from within the Property Editor:

1. Load an object into the Property Editor.
2. Switch to the **Positioning** tab.
3. Click the **Selector** button (>>) and select **New** from the menu.

The **New Attenuation** dialog box opens.

4. Select the work unit in which you want to create the attenuation ShareSet.
5. Type a name for the ShareSet and click **OK**.

The new ShareSet is created, and is applied to the current object.

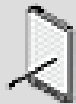
Related Topics

- [Deleting an Attenuation ShareSet](#)

- [Applying an Attenuation ShareSet to an Object](#)
- [Converting Attenuation ShareSets into Custom Instances](#)

Deleting an Attenuation ShareSet

You can delete an attenuation ShareSet if you no longer need it. Before deleting a ShareSet, you should make sure that objects are not still using it. If you delete a ShareSet, it is automatically removed from all objects that subscribe to it.



Note

The Shared by field in the Attenuation Editor displays a complete list of objects using the current attenuation ShareSet.

To delete an attenuation ShareSet:

1. In the Project Explorer, switch to the ShareSets tab.
2. In the Attenuation section, click the ShareSet you want to delete.
3. Press the **Delete** key.

The ShareSet is deleted and removed from all objects that subscribe to it.

Related Topics

- [Creating an Attenuation ShareSet](#)
- [Applying an Attenuation ShareSet to an Object](#)
- [Converting Attenuation ShareSets into Custom Instances](#)

Applying Attenuation Instances to Objects

After you have created the attenuation ShareSets for your project, you can start applying them to your objects. These instances can be either custom or shared, depending on whether you want their properties to affect one object or several.

When applying attenuation instances, you can do any of the following:

- [Applying an Attenuation ShareSet to an Object](#)
- [Converting Attenuation ShareSets into Custom Instances](#)

Applying an Attenuation ShareSet to an Object

If you want to attenuate sound or music objects, you can apply an attenuation ShareSet to the object. Several objects can subscribe to the same ShareSet, which can save you a great deal of time when changes are required.

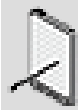
To apply an attenuation ShareSet to an object:

1. Load an object into the Property Editor.
2. Switch to the Positioning tab.
3. In the Attenuation group box, click the **Selector** button (>>).

The list of attenuation ShareSets is displayed.

4. Select the attenuation ShareSet you want to apply.

The name of the ShareSet appears in the corresponding text box signifying that the ShareSet has been applied to the object.



Note

To edit the attenuation properties, click the **Edit** button.

Related Topics

- [Copying Settings from One Platform to Another](#)
- [Creating an Attenuation ShareSet](#)
- [Deleting an Attenuation ShareSet](#)
- [Converting Attenuation ShareSets into Custom Instances](#)

Converting Attenuation ShareSets into Custom Instances

Attenuation instances in Wwise fall into two categories:

- **Custom attenuation instances** can be applied to any one sound or music object in the hierarchy. When you change the properties of a custom instance, only this object is affected.
- **Attenuation ShareSets** can be applied to many objects in your project hierarchy. When you change the properties of a ShareSet, all the objects using the ShareSet are affected.

By default, all attenuation instances you create begin as ShareSets. However, if you prefer, you can convert an attenuation ShareSet into a custom attenuation instance. After the change is made, all modifications to the attenuation instance will apply only to the object that subscribes to it.

To apply an attenuation instance to an object:

1. Load an object into the Property Editor.
2. Switch to the Positioning tab.

3. From the Mode list, select one of the following options:

Use **ShareSets** to apply the ShareSet to the current object.

Define **custom** to apply a custom attenuation instance to the current object.



Note

To edit the attenuation properties, click the **Edit** button.

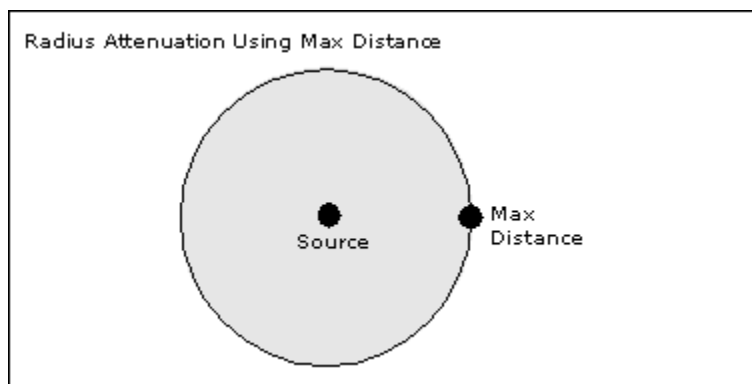
Related Topics

- [Creating an Attenuation ShareSet](#)
- [Deleting an Attenuation ShareSet](#)
- [Applying an Attenuation ShareSet to an Object](#)

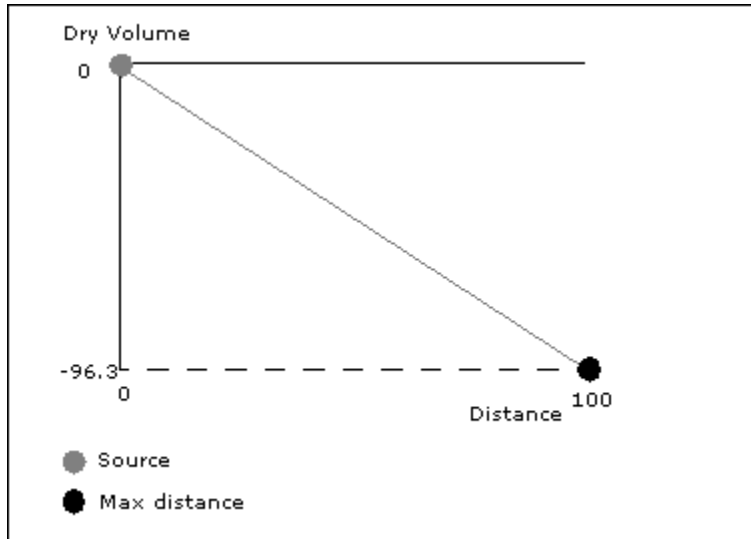
Defining the Attenuation Curves for Various Object Properties

To create the attenuation of objects in Wwise, you can create a series of curves that define a relationship between certain properties in Wwise, such as volume and low pass filter, and the distance the point source is from the listener in game.

Each curve uses a max distance value that defines the point where maximum attenuation of the sound, music or motion object occurs. Since sounds and motion emanate from an omni-directional source, the max distance value creates a spherical radius around each source.

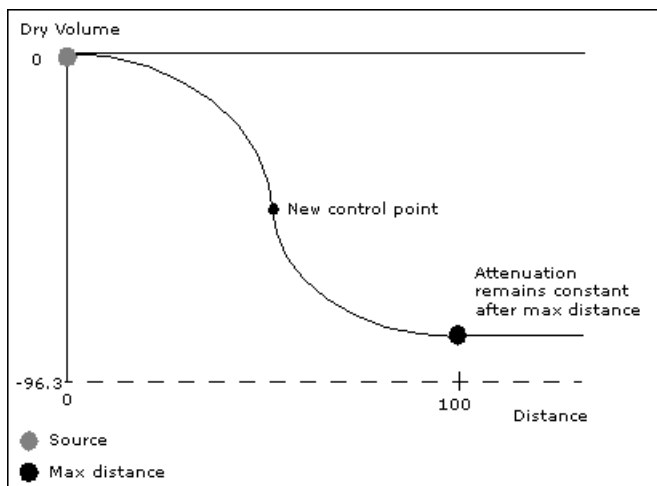


By default, the attenuation of your object will be applied using a linear interpolation from the source to the max distance.



Although this may work in most cases, specific sounds or motion objects may require more advanced curves. To have more control over the attenuation curve, you can add control points. These points break up the attenuation curve so that you can better manage the attenuation of your sound or motion object.

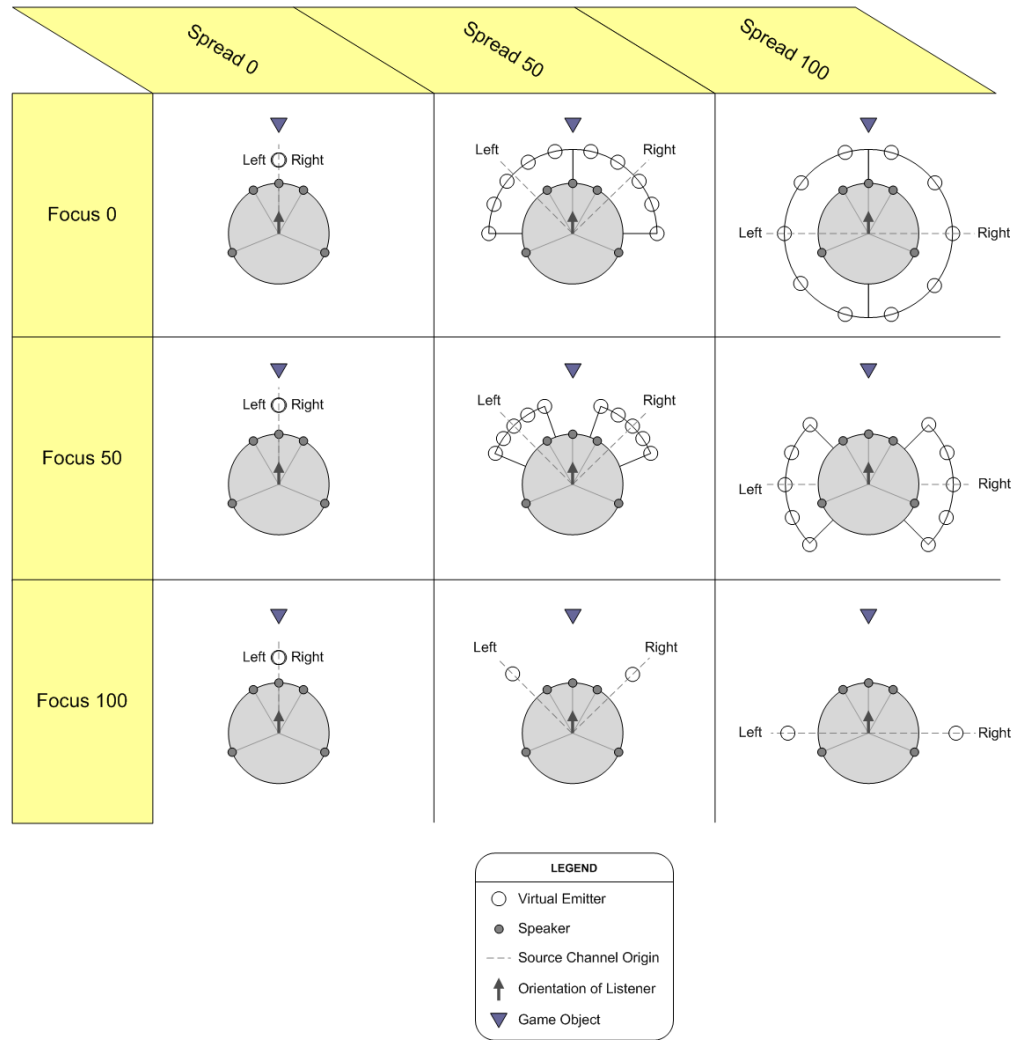
To create a more detailed and complex positioning curve, you can also define the shape of each curve segment. A curve segment is any part of the curve between two control points. You can choose from a variety of curve shapes, including linear, constant, logarithmic, exponential, and s-curve. For more information on specifying curve shapes and other information about working in the graph view, refer to [Chapter 42, *Getting to Know the Graph View*](#).



You can create attenuation curves for the following Wwise properties:

- **Output Bus Volume** - The attenuation or amplitude of the signal routed to the audio output bus.

- **Auxiliary Send volumes** - The attenuation or amplitude of the signal sent to game-defined and user-defined auxiliary busses.
- **Low-pass filter** - The recursive filter that attenuates high frequencies based on a specified value. The units for the low-pass filter represent the percentage of low pass filtering that has been applied, where 0 means no low-pass filtering (signal unaffected) and 100 means maximal attenuation.
- **High-pass filter** - The recursive filter that attenuates low frequencies based on a specified value. The units for the high-pass filter represent the percentage of high-pass filtering that has been applied, where 0 means no high-pass filtering (signal unaffected) and 100 means maximal attenuation.
- **Spread** - The amount or percentage of audio that is spread to neighboring speakers allowing for sounds objects to change over distance from a point source at low values to a completely diffused propagation at high values. A value of 0 means that the channels of an emitting source positioned next to a speaker will only be played in that speaker. A value of 100 means that the channels of the emitting source will be diffused so that they are heard or felt through all speakers.
- **Focus** - The percentage value is used to condense the virtual emitters generated by the spread value. For a focus of 0%, the virtual emitters remain unchanged, but at higher values each virtual points are moved closer around the source channel origin.



Note

The Auxiliary send volumes, and Low pass filter curves have no impact on motion FX objects.

Attenuation property values are relative, which means that the attenuation value is added to the existing property values of the associated object.

To define the attenuation curves:

1. Load an object into the Property Editor and switch to the Positioning tab.
2. In the Attenuation group box, click **Edit**.

The Attenuation Editor opens with the property settings of the selected attenuation instance.

3. In the Max distance text box, specify the distance from the source point where the sound will reach its maximum attenuation.

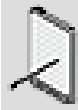


Note

After the max distance value, the attenuation settings remain constant.

4. In the Curves group box, select the **Output Bus Volume** curve from the list.

The default **Output Bus Volume** curve is displayed in the graph view.



Note

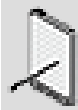
The first point on the curve always represents the point source and the last point on the curve always represents the max distance value.

5. Manipulate the **Output Bus Volume** attenuation curve, by doing any of the following:

Add points along the curve

Drag the points to a new location or type specific values into the X and Y coordinate boxes.

Define the shape of each curve segment.



Note

For specific information on zooming or panning in the graph view, displaying several curves simultaneously, adding, moving, or deleting points, specifying the scaling method, or changing the shape of curve segments, refer to [Chapter 42, *Getting to Know the Graph View*](#).

6. For the remaining curves, select one of the following options from the Curve list:

None to not use an attenuation curve for the corresponding property. When “None” is selected, the corresponding property is NOT attenuated and remains at its full strength.

Use **Output Bus Volume** to use the same **Output Bus Volume** curve for the corresponding property. This option is only available for the *Auxiliary send volumes* curve.

Custom to create a custom attenuation curve for the corresponding property.



Note

For motion FX objects, the Auxiliary Send Volumes and Low pass filter curves should be set to None, because these properties do no impact these types of objects.

Related Topics

- [Copying Settings from One Platform to Another](#)
- [Working with 3D Sound, Music, and Motion FX Objects](#)
- [Applying Distance-Based Attenuation](#)
- [Defining the Attenuation Curves for Various Object Properties](#)
- [Simulating Sound and Motion Directivity Using Cone-Shaped Boundaries](#)
- [Routing Audio Signals to the Center Speaker](#)
- [Zooming and Panning the Graph View](#)
- [Adding Control Points](#)
- [Defining the Scaling Method of the Graph View](#)
- [Specifying the Shape of the Curve Between Control Points](#)

Simulating Sound and Motion Directivity Using Cone-Shaped Boundaries

By default, sounds in Wwise emanate from an omni-directional source. In reality, however, sounds usually have some direction to them. To simulate the direction of a sound in Wwise, you can use a sound cone. The sound cone simulates the propagation of the sound in a particular direction using angles of varying degrees. As the listener moves outside these angles, the output bus volume (acting as the dry component) is attenuated. Because 3D game-defined positioning uses live game data, the direction of the sound cone is ultimately controlled by the orientation of the game object.

The following angles define the regions within the sound cone:

- **Inner angle** - The angle that defines the region where no output bus volume attenuation or low pass filter effects occur.
- **Outer angle** - The angle that defines the region where output bus volume attenuation and low pass filter effects remain at their maximum levels.

Output Bus Volume roll-off occurs in the transition area or the region between the borders of the inner and outer angles. The volume is attenuated using linear interpolation between the inner angle border where no attenuation occurs and the outer angle border where the max attenuation value is reached. In the region defined by the outer angle, volume attenuation is always equal to the max attenuation value.



Note

The output bus volume attenuation and low pass filter values within the cone attenuation are added to those defined by the output bus volume and low pass filter attenuation curves. The auxiliary send volumes and values remain unchanged.

The Max Radius Attenuation display located beside the cone attenuation properties gives you a visual representation of the different regions within your sound cone. It automatically updates as you change the inner and outer angle values.

Simulating Direction of Motion

Although motion emanates from an omni-directional source, there may be situations in game where you want to simulate the propagation of the motion in a particular direction. In these situations, you can use the Cone Attenuation properties. The sound cones work in the same way for motion as they do for sounds, except that the Low Pass filter setting has no effect on motion objects. The sound cones use angles of varying degrees to attenuate the motion based on the listener's position in relation to the orientation of the source.

To simulate sound and motion directivity using sound cones:

1. Load an object into the Property Editor and switch to the Positioning tab.
2. In the Attenuation group box, click **Edit**.

The Attenuation Editor opens with the property settings of the selected attenuation instance.

3. Select the **Cone Attenuation** option.

The sound cone controls become available.

4. In the Inner angle text box, specify an angle that defines the region where you want no output bus volume attenuation to occur.
5. In the Outer angle text box, specify an angle that defines the region where output bus volume attenuation and low pass filter effects remain at their maximum levels.



Note

The area between the inner and outer angles is called the transition area. In this area, the output bus volume is attenuated using a linear interpolation between the inner angle border where no attenuation occurs and the outer angle border where the max attenuation value is reached.

6. In the Max attenuation text box, define the amount by which the output bus volume will be attenuated when a sound or motion object falls within the outer angle.
7. To apply a low pass filter to sounds that fall between the inner and outer angles, type a value in the Low pass filter text box. The low pass filter is a recursive filter that attenuates high frequencies based on the value specified.



Note

The units for the low pass filter represent the percentage of low pass filtering that has been applied, where 0 means no low pass filtering (signal unaffected) and 100 means maximal attenuation. The low pass filter value has no effect on motion objects.

Related Topics

- [Working with 3D Sound, Music, and Motion FX Objects](#)
- [Applying Distance-Based Attenuation](#)
- [Defining the Attenuation Curves for Various Object Properties](#)
- [Routing Audio Signals to the Center Speaker](#)

Previewing the Attenuation Settings of a Sound

After you have set up the attenuation properties for your sounds, you can preview your settings directly in Wwise. This will give you an idea of what the game player will hear as they move around within the sound's attenuation radius. By having the ability to preview a sound's attenuation directly in Wwise, you can fine-tune and tweak your settings even before any of your audio has been integrated into the game.

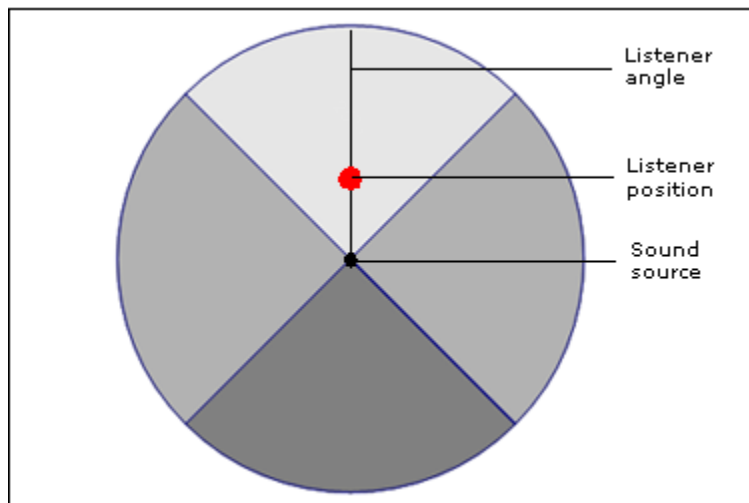


Note

You can only preview the attenuation settings of a sound that uses 3D Game-defined positioning.

To preview your attenuation settings, you need to be able to modify the position of the listener within the sound's attenuation radius. This is done using the Attenuation Preview controls, which are located in the Attenuation Editor. The Attenuation Preview is a graphical representation of a sound's attenuation radius, which is defined by the Max distance value. The source of the sound is always located directly in the middle of the circle. You can modify the position of the listener in relation to the sound source using the following two controls:

- A **small red circle** - Represents the actual position of the listener in relation to the sound source. This small red circle can be moved freely anywhere within the attenuation radius. As you move the red circle away from the center, the volume of the sound will decrease until it reaches maximum attenuation at the outer edges of the circle. By specifying a position for the listener, you automatically determine the listener's distance from the sound source.
- A **thin black line** - Represents the angle at which the listener is located in relation to the sound source. By default, the listener is located directly in front of the sound source (0 degree angle), but you can modify the angle by dragging the thin black line in a circular motion. You can simulate movement around the sound source by modifying the angle a full 360 degrees (-180 to 180).



Both the angle and position controls can be modified during playback to simulate movement of the listener in relation to the sound.

You can reset the position and angle of the listener, by Ctrl+clicking anywhere within the attenuation radius or by clicking the Reset Selector (>>) button in the Transport Control and then selecting Reset Position.



Note

You can also modify the position of the listener in relation to the sound source by dragging the Distance cursor in the graph view.

To preview a sound's attenuation settings:

1. Load an object into the Property Editor that uses 3D Game-defined positioning.
2. Switch to the Positioning tab and click **Edit** in the Attenuation group box.

The Attenuation Editor opens.

3. Click the **Play** icon to begin playing back the sound.
4. To modify the position of the listener, drag the small red circle anywhere within the attenuation radius.

It will sound as if the listener is moving around the sound source. The level of attenuation will change as you move closer to or further away from the source.



Note

By specifying a position for the listener, you automatically define the listener's distance from the sound source. You can also drag the Distance cursor in the graph view to move the listener closer or further away from the sound source.

5. To modify just the angle of the listener and not the distance, do one of the following:

Click anywhere within the attenuation radius.

Drag the thin black line in a circular motion.

It will sound as if the listener is moving around the sound source. If cone attenuation is on, the attenuation will change as the listener moves through the different areas of the sound cone. When modifying the angle, the distance does not change. As a result, the sound will not be attenuated based on distance.



Note

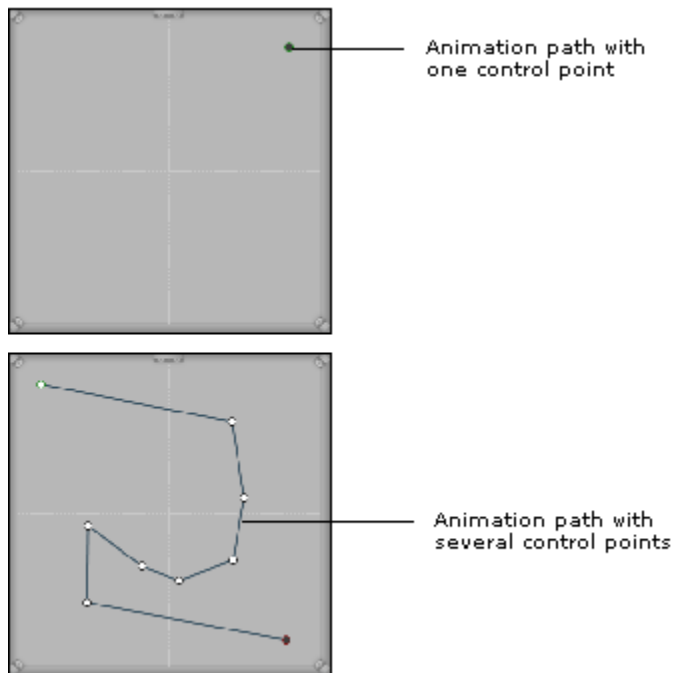
To reset the position and angle of the listener, Ctrl+click anywhere within the maximum attenuation radius or click the Reset Selector (>>) button in the Transport Control and then select Reset Position.

Defining Spatial Positioning Using Animation Paths

If you want to create a very specific experience for your players, you can define the spatial position of your 3D sounds and motion objects using user-defined positioning. When you pre-define the positioning in Wwise, the listener will experience the following regardless of the listener's position and orientation in game:

- Sounds will be heard through the same speakers.
- Motion will be felt through the same motors.

The positioning information is defined using an animation path. Animation paths consist of one or more control points that define the location of the source at any one time. When several points are created, the sound or motion object is animated along the path over time.



User-defined Spatial Positioning - Example

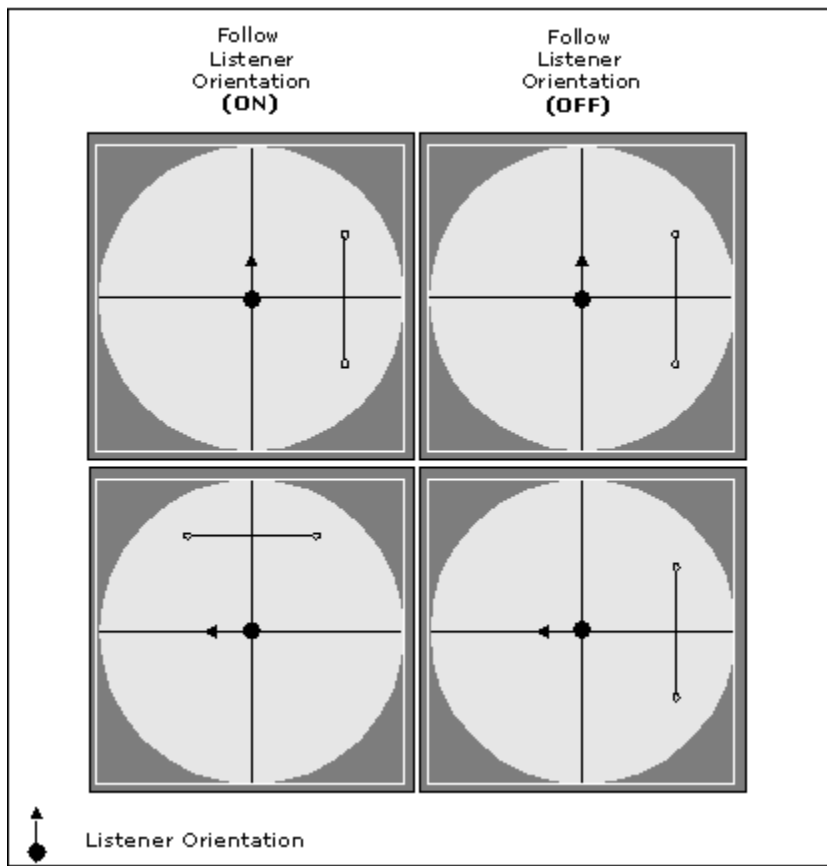
Let's say you are creating a role-playing game that takes place on a ship. As part of the ambience, you want seagull sounds. You want the seagulls to appear as though they are flying around the ship. To accomplish this, you can set the positioning to 3D User-defined and then create several different sound paths that will simulate the flight pattern of the birds around the ship. You can make the seagull sounds even more realistic by attenuating the sound using an attenuation ShareSet.

User-defined positioning can be used for a variety of other purposes in your game, including non-localized ambient sounds, such as insects, birds, and monkeys in a jungle environment.

Creating Animation Paths that Follow the Orientation of the Listener

When using User-defined positioning, you have the option of locking the position of your animation paths to the orientation of the listener in game. When the two are locked, the sound will always be heard through the same speakers regardless of the orientation of the listener. When the two are not locked, the listener moves independently of the path. This means that the sound will be heard through different speakers as the listener turns around.

The following illustration demonstrates what happens to an animation path as the listener changes orientation, when the Follow Listener Orientation option is turned on and when it is turned off.



You can set the Follow Listener Orientation option on the Positioning tab of the object's Property Editor. Since you can't change the listener's orientation in Wwise, you will only be able to hear the difference in a SoundFrame implementation of your level editor or in game.

Working with Animation Paths

You can define the positioning of a sound or motion object by using one source point or by using several control points to create a path. When you create a path, the sound or motion object will be animated along that path over a specified period of time. To give you added power and flexibility, you can create many paths for the same object, modify each of their durations, and then play them back according to a specific sequence, or in a completely random order.

Animation paths are created using the graph view and timeline. For specific information on moving, deleting points, or zooming, panning in the graph view and timeline, refer to [Chapter 42, Getting to Know the Graph View](#).

Creating an Animation Path

Before you can define the positioning of a sound, music, or motion object, you must create an animation path. An animation path is made of one or more

control points. When several points are used, an animation path is created along which the object will travel over a specified period of time.

These paths are created in a graph view. The graph view displays the location of each control point as a percentage of the max distance value. This means that the values for each control point will fall between 0 and +/- 100 depending on which quadrant of the graph it is located in.



Note

When you first create a path, Wwise automatically names it "ObjectName_01_Path". You can rename the path at any time to give it a more meaningful name. Each path, however, must have a unique name.

To create an animation path:

1. In the Position Editor (3D User-defined), click New.

A new animation path is added to the list, and a single control point is added to both the graph view and timeline.

2. In the Path Name text box, type a representative name.
3. Position the first point of your path by dragging it anywhere within the graph view.
4. To add additional points, double-click in the graph view.

The points in the graph view are automatically connected to form an animation path.



Tip

You can fine-tune the position of any point by dragging it in the graph view or by typing values directly in the X and Y coordinate text boxes. You can also change the timing between points by moving them in the timeline.

Related Topics

- [Randomizing the Position of Each Point Along a Path](#)
- [Changing the Path Duration](#)
- [Displaying the Attenuation Radius and Cone in the Graph View](#)
- [Reordering the Animation Path List](#)

- [Deleting an Animation Path](#)
- [Determining How Animation Paths are Played Back](#)

Randomizing the Position of Each Point Along a Path

To add realism and avoid repetition in your game, you can easily randomize the position of each point along each of your paths. To do so, you simply need to define a range of possible values for each point, in both the horizontal and vertical planes. At runtime, Wwise picks a random value from within these ranges to specify the actual position of the control point.



Note

The position of each control point in the graph view is measured as a percentage of the max distance value. Since the Random Range values modify the X, Y, and Z coordinates of these points, they are measured in the same units.

To randomize the position of each point along a path:

1. In the Position Editor (3D User-defined), select a path from the path list.
2. In the Random Range group box, specify a value for the following two properties:

Horizontal - Acts as a +/- offset to the X and Y coordinate values of each point, creating a range of possible values. At runtime, Wwise chooses a value from within this range to define the horizontal position of the control point.

Vertical - Acts as a +/- offset to the Z coordinate value of each point, creating a range of possible values. At runtime, Wwise chooses a value from within this range to define the vertical position of the control point.



Note

The default vertical position (Z value) of each point is 0.

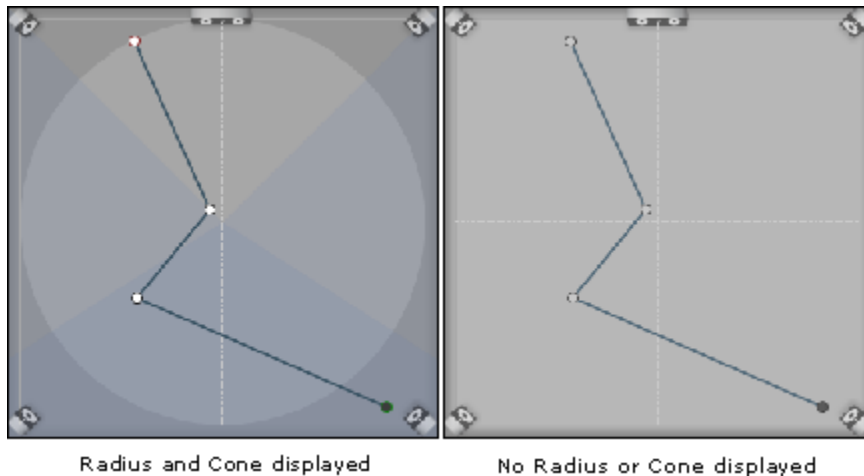
Related Topics

- [Creating an Animation Path](#)
- [Changing the Path Duration](#)
- [Displaying the Attenuation Radius and Cone in the Graph View](#)
- [Reordering the Animation Path List](#)
- [Deleting an Animation Path](#)

- [Determining How Animation Paths are Played Back](#)

Displaying the Attenuation Radius and Cone in the Graph View

If you are using attenuation to simulate the natural roll-off of an audio or motion signal as it moves away from the listener, you may want to use the max distance radius and sound cone as a reference so you can see the different areas of attenuation when creating your paths in the graph view.

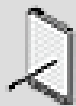


To change the display of the graph view:

1. In the Display Option group box, select any of the following options:

Show Radius to display a representation of the attenuation max distance radius in the graph view.

Show Cone to display a representation of the cone angles (inner and outer) in the graph view.



Note

The Show Radius and Show Cone options are only available if an attenuation instance has been applied to the current object.

Related Topics

- [Creating an Animation Path](#)
- [Randomizing the Position of Each Point Along a Path](#)
- [Reordering the Animation Path List](#)
- [Deleting an Animation Path](#)
- [Determining How Animation Paths are Played Back](#)

Changing the Path Duration

Each path that you create can have a different duration. To change the duration of a path, you must change the length of the timeline. If the existing points along the timeline are not spaced at even intervals (in non-linear mode) then you must specify whether you want the points to be stretched proportionally across the timeline or to keep the points at their existing position.

To change the duration of an animation path:

1. In the Path list, click the path whose duration you want to change.
2. Click the **Configure Timeline** button.

The Configure Timeline dialog box opens.

3. In the Length text box, specify a new duration for the selected path.
4. If the timeline already contains controls points and is in non-linear mode, you must select one of the following options:

Stretch proportionally to reposition the existing control points while maintaining their relative positions to one another on the timeline.

Preserve key values to keep the control points at their existing positions on the timeline. Control points that exceed the length of the new timeline will be deleted.

5. Click **OK**.

The length of the animation path is updated. The timeline also changes to reflect the new duration.

Related Topics

- [Creating an Animation Path](#)
- [Randomizing the Position of Each Point Along a Path](#)
- [Reordering the Animation Path List](#)
- [Deleting an Animation Path](#)
- [Determining How Animation Paths are Played Back](#)

Reordering the Animation Path List

You can reorder the paths in the animation path list if you want them to appear or playback in a particular order.

To reorder the path list:

1. In the path list, click the path that you want to move.
2. Drag the selected path to its new location.



Tip

When moving paths within the path list, a red line appears to help you know where in the list the path will be dropped.

Related Topics

- [Creating an Animation Path](#)
- [Randomizing the Position of Each Point Along a Path](#)
- [Changing the Path Duration](#)
- [Deleting an Animation Path](#)
- [Determining How Animation Paths are Played Back](#)

Deleting an Animation Path

If you no longer need a path, you can remove it from the list.

To delete an animation path:

1. In the Path list, click the path that you want to delete.
2. Click **Delete**.

A confirmation message box is displayed.

3. Click **OK**.

The path is removed from the list.

Related Topics

- [Creating an Animation Path](#)
- [Randomizing the Position of Each Point Along a Path](#)
- [Changing the Path Duration](#)
- [Reordering the Animation Path List](#)
- [Determining How Animation Paths are Played Back](#)

Determining How Animation Paths are Played Back

After creating your paths, you must decide how they will be played back. Like containers, you can decide to play back the paths that you have created randomly or following the order specified in the path list. You can also decide to play all paths one after the other or only one path each time the object is played.

To determine how animation paths are played back:

1. In the Position Editor (3D User-defined), select one of the following options as the play type:

Sequence to play the paths in a sequential order from the start of the list to the end.

Random to play the paths in a random order until all paths have been played.

2. In the Play Mode group box, select one of the following options:

Continuous to play the paths one after the other until all paths have been played.

Step to play a single path from the list each time the object is played.

3. If you selected Continuous, the Loop and Transition time options become available.

To play the entire animation path list indefinitely, select the **Loop** option.

To add a linear transition between the end of one path and the start of another, select the **Transition time** option and then specify the amount of time for the transition in the corresponding text box.

4. If you selected Step, the Pick new path when sound starts option becomes available.

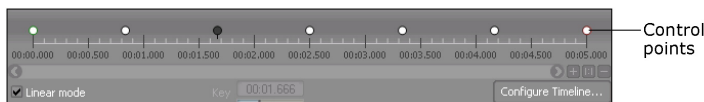
Select the **Pick new path when sound starts** option to force Wwise to use a new path each time a new sound is played regardless of whether the sound is triggered by a play event. This option is particularly useful when you want a different path to be used for each sound within a continuous container.

Related Topics

- [Creating an Animation Path](#)
- [Randomizing the Position of Each Point Along a Path](#)
- [Changing the Path Duration](#)
- [Reordering the Animation Path List](#)
- [Deleting an Animation Path](#)

Working with the Points in the Timeline

When creating animation paths, you need to define the time it will take for the sound to travel along the path. The Position Editor (3D User-defined) timeline allows you to specify where each control point will fall over time.



You can move the control points along the timeline to define how fast sound or motion FX objects will travel between points along the path. You can move and delete points, but you can't add points directly on the timeline. Control points can only be added in the graph view. You can also select several points at the same time to move or delete a complete section altogether.



Note

You can zoom and pan in the timeline to help position control points more accurately.

Changing the Timeline Mode

The timeline in the Position Editor (3D User-defined) can be in one of two modes:

- **Linear** - Control points are automatically placed at even intervals along the timeline.
- **Non-linear** - Control points are placed anywhere on the timeline.

Linear mode produces a constant motion from one key to the next as a sound or motion FX object travels along the path. Non-linear mode, on the other hand, can be used to speed up and slow down an object as it travels between points along the path. Linear is the default timeline mode.

You can further configure certain behaviors of the timeline by clicking the Configure Timeline button. For more information, refer to [Configuring the Positioning Timeline](#).

To change the timeline mode:

1. In the Position Editor (3D User-defined), do one of the following:

Clear the **Linear** option to set the timeline mode to non-linear.

Select the **Linear** option to set the timeline mode to linear.

Selecting Control Points in the Timeline

Before you can move or delete a control point in the timeline, you must select it first. You can select one, several, or all control points at a time.

To select a control point in the timeline:

1. In the timeline, click a control point to select it.

The selected control point turns black.

To select several control points in the timeline:

1. In the timeline, drag a rectangle over the points you want to select.

The selected points turn black.



Tip

You can also select several points by Ctrl+clicking each point.

To select all control points in the timeline:

1. Click anywhere in the timeline to make it active.
2. Press Ctrl+A.

All points in the timeline are selected and turn black.

Related Topics

- [Moving Control Points in the Timeline](#)
- [Deleting Control Points in the Timeline](#)

Moving Control Points in the Timeline

When the timeline is in non-linear mode, you can move control points to change the time between each point. This creates the effect that the sound or motion FX object is speeding up or slowing down between points. A control point can't be moved past the control point before or after it.

To move control points in the timeline:

1. In the timeline, select one or more control points.

The selected control points turn black.

2. Drag the point(s) to their new position on the timeline.



Note

You can also move points by typing a value directly into the Key field. If more than one point is selected, the positive (+) or negative (-) value you type in the Key field will offset the points to the right or left of their original position.

Related Topics

- [Selecting Control Points in the Timeline](#)

- [Deleting Control Points in the Timeline](#)

Deleting Control Points in the Timeline

You can delete any control point in the timeline except the first one.

To delete control points from the curve:

1. Select one or more points in the timeline.

The selected control point(s) turn black.

2. Press the **Delete** key.

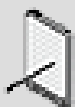
The control point(s) are removed from both the timeline and graph view.

Related Topics

- [Selecting Control Points in the Timeline](#)
- [Moving Control Points in the Timeline](#)

Routing Audio Signals to the Center Speaker

For the sounds in your game that are crucial to game play, you may want to ensure that they are clearly heard by the game player. To achieve this, you can route any percentage of a 2D or 3D sound or music object to the center speaker. For example, the voices of the drivers or announcers at a race track can pass completely through the center speaker so that they are perfectly audible no matter how loud the other sounds are in the game.



Note

The Center % option is not supported on the Wii platform and has no impact on motion FX objects.

Center % Settings

The following table describes how much of the signal will be routed to the center speaker at various Center % settings:

Setting	Center %
0%	Creates a phantom center image - no signal is passed to the center speaker.
1-99%	A varying amount of signal is passed through the center speaker. As the Center % is increased, the levels of the front left and front right speakers decrease.

Setting	Center %
100%	Creates a discrete center image - full signal is passed to the center speaker.



Note

The amount of the signal that is spread to other speakers is based on the position of the source in relation to the different speakers.

To route audio signals to the center speaker:

1. Load a top-level parent object into the Property Editor.
2. Switch to the Positioning tab.



Note

If the object is not a top-level object, you must select the Override parent option before you can set the Positioning options.

3. In the Center % text box, specify the percentage of volume that you want to pass through the center speaker.

Related Topics

- [Working with 2D Sound, Music, and Motion FX Objects](#)
- [Working with 3D Sound, Music, and Motion FX Objects](#)
- [Applying Distance-Based Attenuation](#)
- [Defining Spatial Positioning Using Animation Paths](#)

Positioning Tips and Best Practices

Before defining the positioning for your objects in Wwise, you may want to review the following sections, which provide you with a series of examples, tips, and best practices that can help you better manage the positioning of your sounds, music, and motion FX objects in game.

Positioning - Example (Part 2)

Now that the different positioning options available in Wwise have been described in detail, let's see how the different options can be used to define the positioning for the sounds and motion effects in our first-person [Positioning - Example \(Part 1\)](#).

- **Footsteps** - Since this is a first-person game, the footstep sounds of the main character will always be attached to the camera. Since there is no movement

and no attenuation for these sounds, basic 2D positioning is appropriate in this case. For the other agents, however, you will need to match the footstep sounds to their movement by attaching the sounds to the “agent” game objects. 3D Game-defined positioning with spatialization would be appropriate in this case.

- **The torches that light up the enemy's jungle base** - These sounds will be attached to the torch game objects. Although they are fixed in one place, the location of the sound emitter and its distance from the microphone will change as the player moves. To simulate this type of sound, you can use 3D Game-defined positioning with both spatialization and attenuation.
- **A group of terrorists talking in a hut** - These sounds will be attached to the terrorist game objects, which can move freely within the game environment. To simulate this type of sound, you could use 3D Game-defined positioning with both spatialization and attenuation.
- **A mosquito buzzing overhead** - In this example, the mosquito can be heard buzzing around, but cannot be seen. Since the sound emitter must move within the 3D space, but no actual game object exists, 3D User-defined positioning would be appropriate in this case. A series of randomly played back sound paths using both spatialization and attenuation can create very realistic insect sounds.
- **Updates received from headquarters** - The communication received from headquarters is not associated with any particular game object and does not move within the surround environment, therefore, 2D positioning would be appropriate in this case. Since the updates are crucial to the mission, you may also want to route some or all of these sounds through the center speaker.
- **The whispered communication between special agents on this mission** - The teammates whispered voices will be attached to their respective game objects, so 3D Game-defined positioning would be appropriate for these sounds. The agents will be moving around one another requiring some kind of spatialization, but since the agents must work together as a team, the communication between them will not require any attenuation. The communication between teammates is crucial to the mission, so you may also want to route some or all of these sounds through the center speaker.
- **The detonation of explosives used to destroy the base after the mission has been successfully completed** - The detonation of the explosives will be heard and felt by the operatives. These sound and motion FX objects will be attached to the explosives game objects. Although they are fixed in one place, the location of the sound/motion emitter and its distance from the listener will change as the player moves. To simulate this type of sound and motion effect, you can use 3D Game-defined positioning with both spatialization and attenuation.
- **The constant rumbling of the island's volcano** - The rumbling of the volcano is a constant sound and motion effect on this remote island. Both the sound and motion object would most likely be attached to the “island” game object.

Some attenuation would make the rumbling appear louder or more intense as the players move closer to the island. Since there is no movement to the sound or motion, spatialization may not be necessary in this case.

- **The interactive music** - Since the music is not associated with any particular game object and requires no movement within the surround environment, 2D positioning would be appropriate. For our example, we want to pan some of the music tracks so that the music is balanced between the front and rear speakers.

Refer to the following table for a complete overview of the positioning options that could be used to create the different sounds in this example.

Sound	2D		3D			
	No Panning	Panning	Game-Defined	User-Defined	Spatialization	Attenuation
Agent's footsteps	✓					
Torches			✓		✓	✓
Terrorists talking			✓		✓	✓
Mosquito buzzing				✓		
Updates from HQ	✓					
Agent communication			✓		✓	
Explosions			✓		✓	✓
Rumbling of volcano			✓			✓
Interactive music		✓				

This example describes one way to create different types of positioning and propagation using the different options available in Wwise. The options you choose will depend on the sounds and motion effects themselves, the game you are creating, and the specific effect you are trying to create.

Performance Optimizations

- Use mono sounds when not using spread attenuation curve - if you are not planning to use the spread curve to widen your audio signal, you should use mono sounds to optimize performance. When spread is not used, all the input channels of a stereo sound will be mapped to the same position and will have to be rendered dynamically, whereas if you use mono sounds, the operation will be done offline and won't take any CPU during game play.
- Re-use or reduce the number of curves in Attenuation Editor to improve performance - keep in mind that the more curves you create in the Attenuation Editor, the more processing power and memory is used. To improve performance, you can either re-use the Output Bus Volume curve (for Auxiliary Send Volumes) or not use a curve at all.

- Use a small number of points and linear curve segments to improve performance - keep in mind that the more points you add along the curve and the more complex the curve shape, the more processing power and memory is used. In most cases, a curve with two or three points using linear segments will be sufficient to get the attenuation results you need.
- Share attenuation property settings using ShareSets - if several of the sound, music, and/or motion FX objects within your game have similar attenuation properties, you can share these property settings using a ShareSet. By sharing the attenuation property settings, you can save on both memory and time to make changes to the attenuation properties.
- Use the Positioning Type RTPC to re-use sounds for similar purposes. For example, the player's footsteps can be set to 2D and the enemies' footsteps set to 3D, to both using the same sound hierarchy. This can save lots of memory in the Default Pool.

Overview of Specific Positioning Scenarios

Let's take a look at some specific scenarios to give you a better understanding of how the different positioning and attenuation settings work in Wwise.



Note

By default, 2D and 3D sounds are not played through the center speaker. To route any portion of a signal through the center speaker, use the Center % property slider.

Scenario 1

- Positioning: 3D
- Attenuation: None
- Spatialization: Off

Result: These settings effectively give you the same positioning as a 2D sound.

Scenario 2

- Positioning: 3D
- Attenuation: Simple linear curve



- Spatialization: Off

Result: These settings effectively give you a sound that will grow quieter as the listener moves away from the sound source (attenuation), but the sound

will always be placed exactly the same as it appears in the original sound asset without any positioning or rotation applied (spatialization).

Scenario 3

- Positioning: 3D
- Attenuation: None
- Spatialization: On

Result: These settings effectively give you a sound that originates from a specific location (spatialization), but the volume never attenuates, no matter how far the listener gets from the sound source (attenuation).

Scenario 4

- Positioning: 3D
- Attenuation: Simple linear curve



- Spread: Simple linear curve



- Spatialization: On
- Sound source: Mono

Result: These settings effectively give you the following:

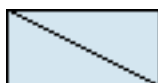
- When the listener is far from the sound source, the sound originates from a specific location (spatialization), is at a reduced volume (attenuation), and is played mostly in one speaker (spread).
- When the listener is close to the sound source, the sound originates from a specific location (spatialization), is nearly at full volume (attenuation), and nearly distributed equally across both speakers (spread).

Scenario 5

- Positioning: 3D
- Attenuation: Simple linear curve



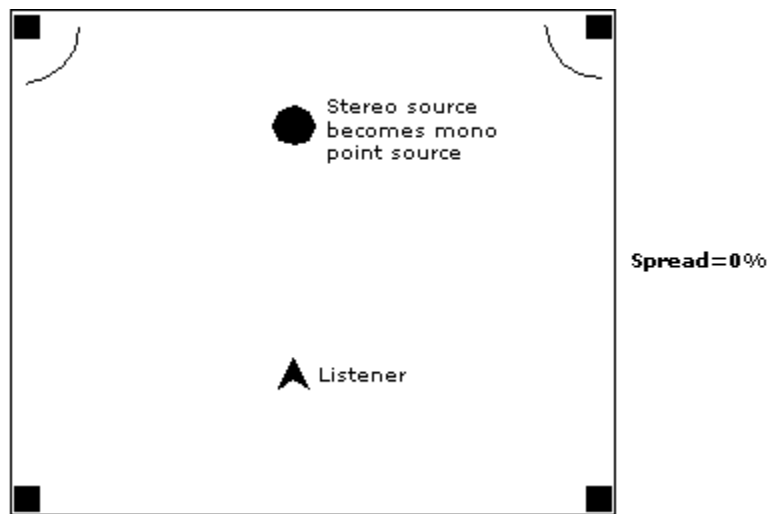
- Spread: Simple linear curve



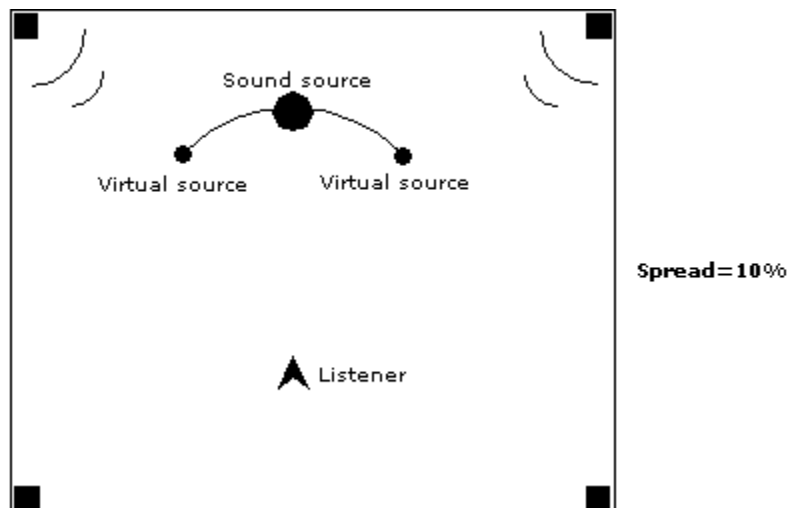
- Spatialization: On
- Sound source: Stereo

Result: These settings effectively give you the following:

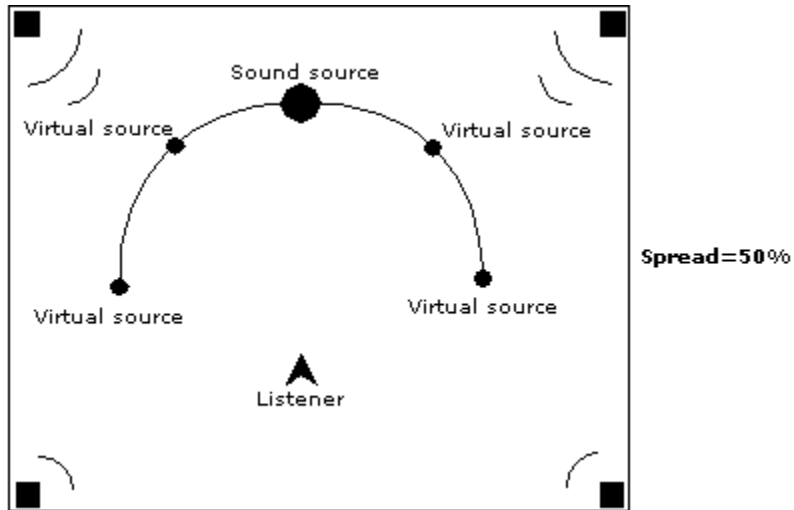
- When the listener is far from the sound source, the sound originates from a specific location (spatialization), and is at a reduced volume (attenuation). For a stereo source that is spatialized without spread, both channels are folded down to create a mono 'point source'. It is for this reason that we recommend using mono files when there is no spread as it is more efficient in terms of CPU.



- When spread is used, new “virtual sources” are defined that are offset from the original source. For example, for small spread values, a virtual source will be computed to the left and to the right of the real position and their contribution will be added to the speakers, in exactly the same way as the normal no-spread sounds, only in a slightly different position.



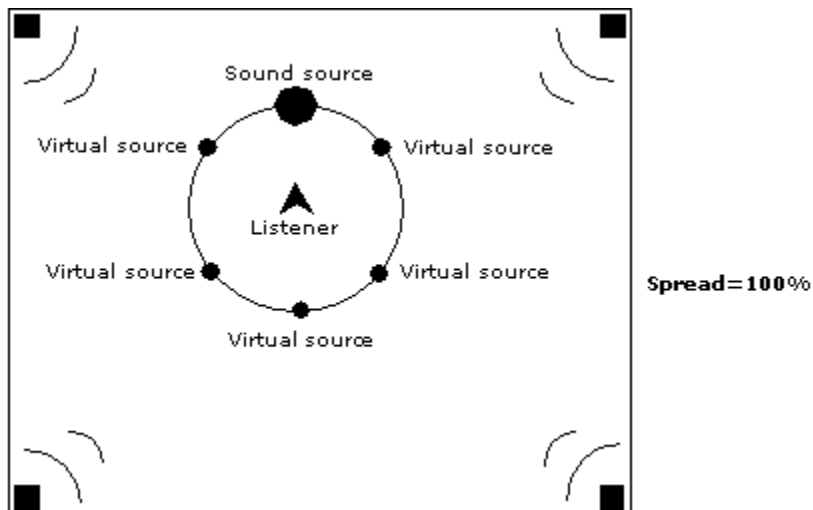
As the spread value increases, there will be more of those virtual sources to cover a larger arc around the listener. Obviously, the power of those sources is lower than the real source to maintain the total power constant.



Note

Note that these sources are used for volume computation only, and no new sounds are actually played.

- When the listener is close to the sound source, the sound originates from a specific location (spatialization), and is nearly at full volume (attenuation). When used with a high spread value, the sound will come from all directions. The left and right channels of a stereo sound will be spread separately.



- Now, the case where distance = 0, requires special attention. In Wwise, all spatialization computations (and cone attenuation) are based on angles. When distance = 0, Wwise can't determine if the listener is facing front, left, right, and so on. You should avoid letting this scenario happen in the context of your game. If such a case does happen during gameplay, Wwise will simply create a mono version of the stereo sound to avoid computing out-of-range volumes. The same logic applies for the cone attenuation. If the orientation of the listener is unknown, Wwise assumes there is no attenuation at all. The same is true for the Cone LPF.

Understanding Channel Configurations

The Wwise pipeline supports multiple channel configurations at different levels depending on the source material, the output device, the bus settings, and so on. Different platforms also have different limitations.

Here are the different channel configurations found in Wwise with the channel names, suggested angles for the speakers and ordering:

1.0 (mono)	
Interface Element	Description
Center (C)	0 degrees

2.0 (stereo)	
Interface Element	Description
Left (L)	22-30 degrees
Right (R)	22-30 degrees

3.0	
Interface Element	Description
Left (L)	22-30 degrees
Right (R)	22-30 degrees
Center (C)	0 degrees

4.0	
Interface Element	Description
Left (L)	22-30 degrees
Right (R)	22-30 degrees
Surround Left (SL)	90-110 degrees
Surround Right (SR)	90-110 degrees

5.1	
Interface Element	Description
Left (L)	22-30 degrees
Right (R)	22-30 degrees

Defining Positioning for Sound and Motion

5.1	
Interface Element	Description
Center (C)	0 degrees
Surround Left (SL)	90-110 degrees
Surround Right (SR)	90-110 degrees
Low Frequency Effects (LFE)	N/A

7.1	
Interface Element	Description
Left (L)	22-30 degrees
Right (R)	22-30 degrees
Center (C)	0 degrees
Surround Left (SL)	90-110 degrees
Surround Right (SR)	90-110 degrees
Back Left (BL)	135-150 degrees
Back Right (BR)	135-150 degrees
Low Frequency Effects (LFE)	N/A

Wwise also supports extended standard channel configurations with "height" speakers. Below is a definition of all available channels, followed by a description of standard configurations available on busses.

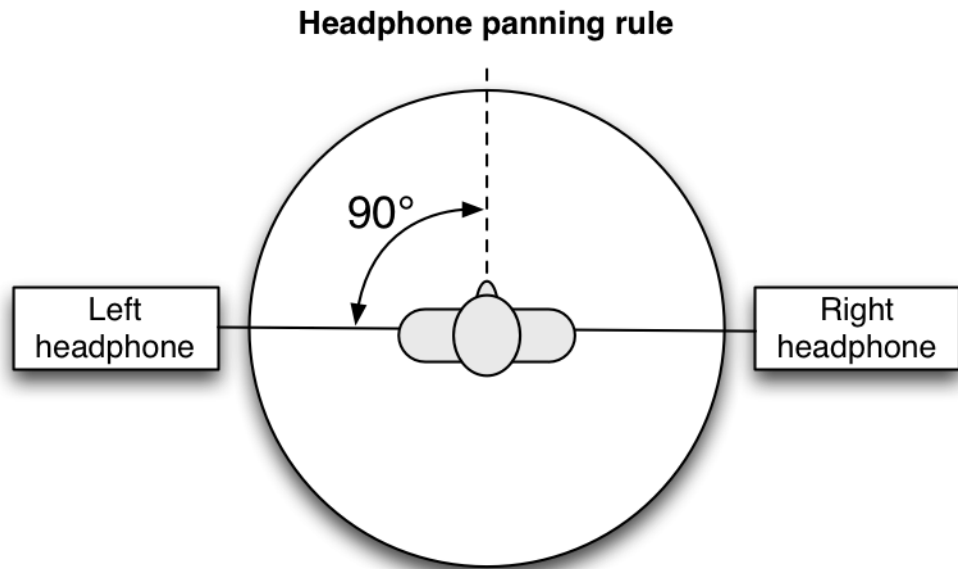
Standard configurations extended with height channels	
Interface Element	Description
Left (L)	22-30 degrees
Right (R)	22-30 degrees
Center (C)	0 degrees
Surround Left (SL)	90-110 degrees
Surround Right (SR)	90-110 degrees
Back Center (BC)	180 degrees
Back Left (BL)	135-150 degrees
Back Right (BR)	135-150 degrees
Top (T)	Straight above
Height Front Left (HFL)	Same as L, above horizon
Height Front Center (HFC)	Same as C, above horizon
Height Front Right (HFR)	Same as R, above horizon
Height Back Left (HBL)	Same as BL (or SL), above horizon
Height Back Center (HBC)	Same as BC, above horizon
Height Back Right (HBR)	Same as BR (or SR), above horizon
Low Frequency Effects (LFE)	N/A

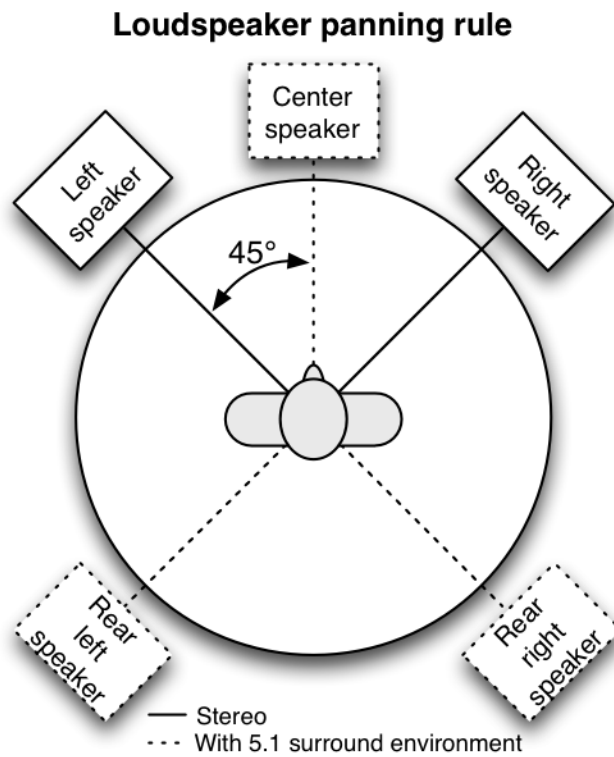
Available bus configurations	
9.1	L-R-C-SL-SR-HFL-HFR-HBL-HBR-LFE

Available bus configurations	
10.1	L-R-C-SL-SR-T-HFL-HFR-HBL-HBR-LFE
11.1	L-R-C-SL-SR-T-HFL-HFC-HFR-HBL-HBR-LFE
13.1	L-R-C-SL-SR-BL-BR-T-HFL-HFC-HFR-HBL-HBR-LFE

Speakers vs Headphones Panning Rules

In Wwise there are two different panning rules: Headphones and Speakers. By default, all platforms use the speaker panning rule with the exception of the handheld consoles that use the headphone panning rule. The difference in between the two modes is subtle but helps to provide a realistic and accurate audio experience depending on your listening set-up. This setting can be auditioned in Wwise but also can also be set in the game at run-time.





To audition the two modes:

- From the menu bar, click **Audio > Stereo Channel Configuration (Speakers)**
- From the menu bar, click **Audio > Stereo Channel Configuration (Headphones)**



Note

You can also set the panning rule in the game, please refer to `AK::SoundEngine::SetPanningRule` in the sound engine documentation.

Chapter 11. Managing the Priority of Sounds and Motion

Overview	311
Understanding How Wwise Prioritizes Sounds and Motion Objects	313
Limiting Object Playback Instances	316
Defining Playback Priority	320
Managing Low-Volume Sounds and Motion Objects	323
Priority Tips and Best Practices	325

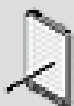
Overview

At any moment in a game, you can have many sounds, music, and motion objects playing at the same time, even to the point where the number of objects surpasses the limit set by the project team. To effectively manage the number of sounds and motion objects that are played, you must determine how many can play simultaneously, and which ones take priority.

In Wwise, there are three main properties that can help you determine which objects will be played at any point in the game:

- **Playback limit** - A limit to the number of sound, music, or motion instances that can be played at any one time (excluding any virtual voices).
- **Playback priority** - The importance of one sound, music, or motion object relative to another.
- **Volume threshold** - A certain volume level below which sound, music, and motion objects are not played.

By setting limits, assigning priorities, and specifying a minimum volume level, you can effectively and creatively manage the many sounds and motion objects in your game while respecting any memory limitations that have been put upon you.

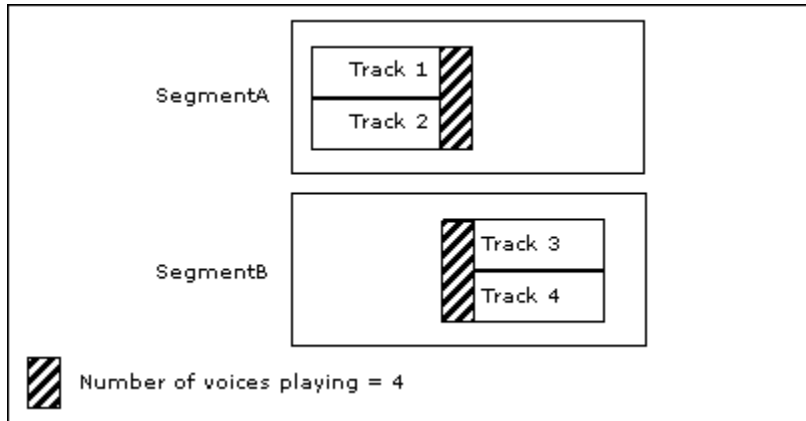


Note

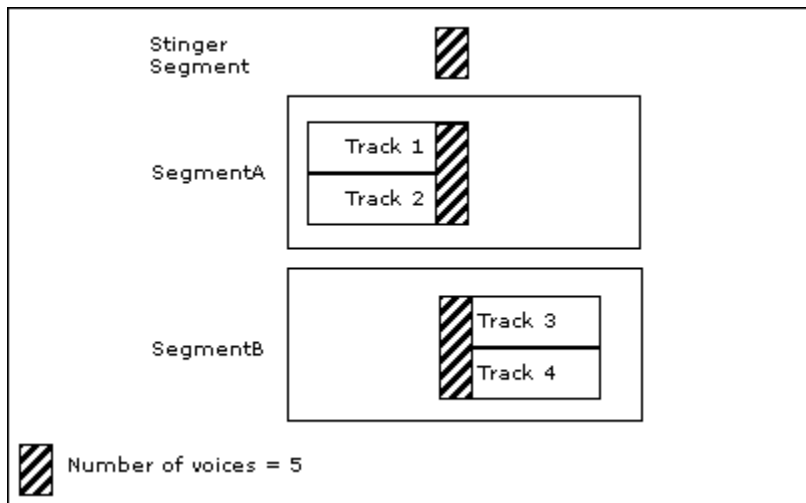
In addition to the three techniques mentioned above, your programmer can also define a memory threshold for one or more of the sound engine's memory pools. When the Memory Threshold is enabled, the sound engine periodically checks that the percentage of memory being used is below the specified threshold. If it goes over the threshold, the sound engine will start dropping sounds with lower priorities to free up space for the higher priority sounds. For more information on the Memory Threshold, refer to the [Wwise SDK documentation](#).

Playback Limit and Priority - Example

Let's say you are composing music for a game that will have very restricted bandwidth. You have been told you can have no more than four voices playing back at once. In this case, you could set the playback limit of your top-level parent object to four. Each of your music segments will contain no more than two tracks, so when the pre-entry and post-exit parts of each track are played during a transition only two voices are taken up. This way, if two segments are playing at the same time, you will still only be within the four voices playback limit.

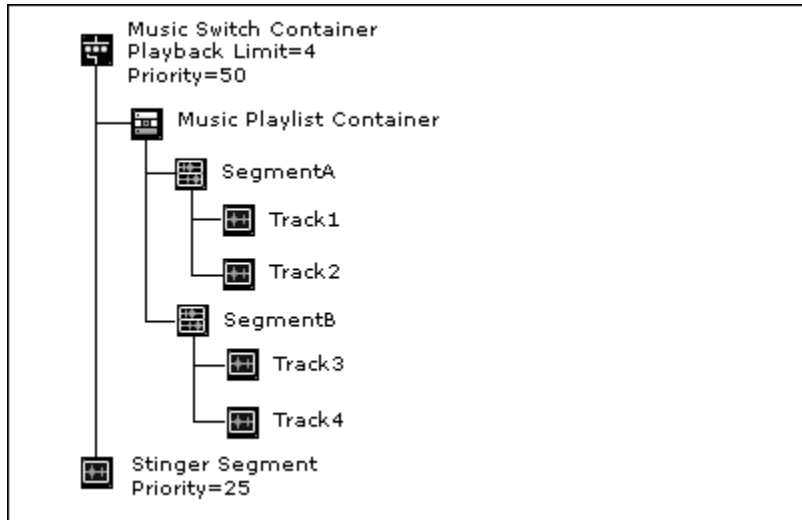


If, however, a stinger is triggered by the game during a transition, you will surpass the maximum of four streams.



When the number of voices surpasses the playback limit, Wwise looks at the playback priority of each music object to see which one to stop playing. In this case, you would probably set the stinger segment to a lower priority to ensure that the “regular” music continues to play. If, however, all five objects in this example had the same priority, you can choose to kill the newest or the oldest instance.

The following illustration shows how you might set up your music hierarchy along with the values you may assign for playback limit and priority in a case like this.



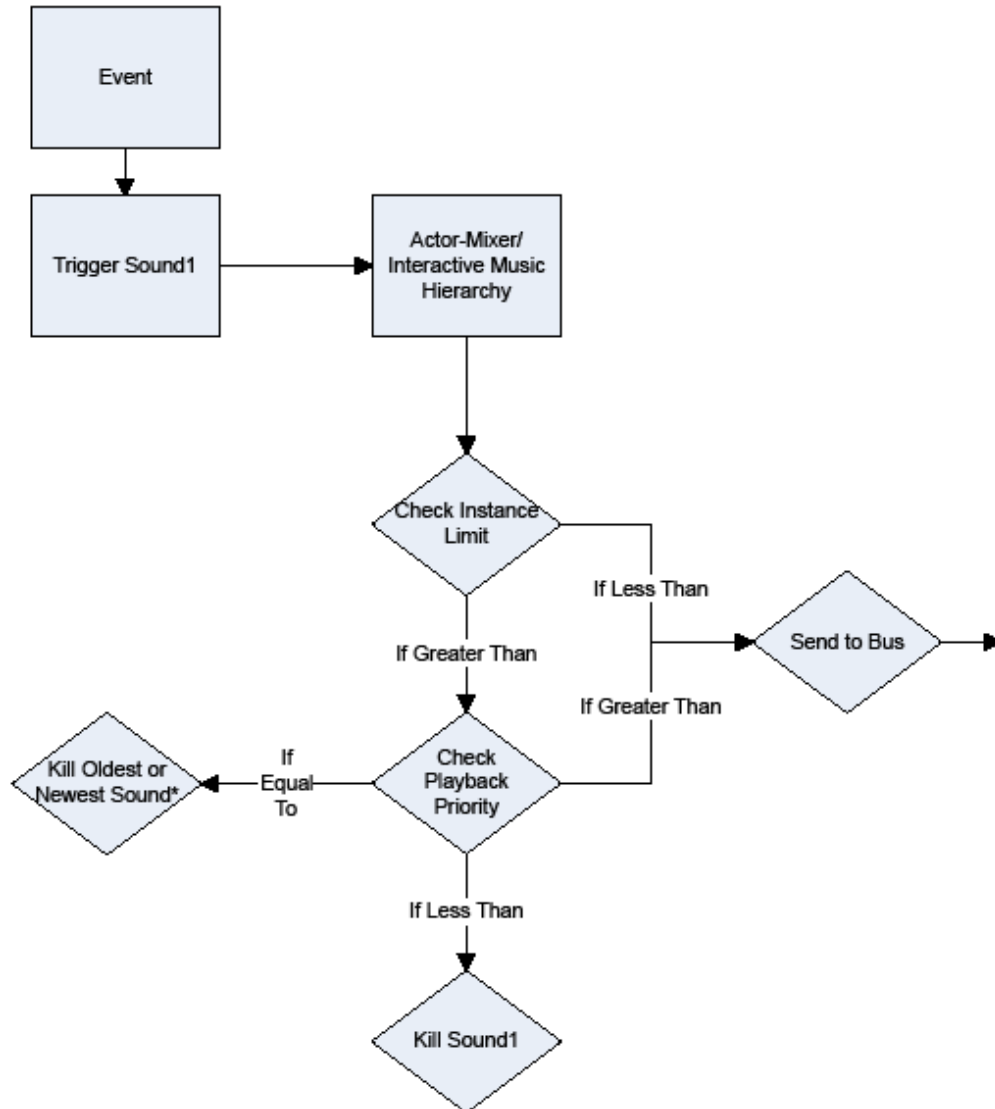
For more information about music objects and building your interactive music hierarchy, refer to [Chapter 22, Building the Interactive Music Hierarchy](#).

Understanding How Wwise Prioritizes Sounds and Motion Objects

The advanced playback settings determine which sounds and motion objects are played at any point in your game. These playback settings are defined at two different levels within Wwise: at the object level in the Actor-Mixer and Interactive Music hierarchies, and at the bus level in the Master-Mixer hierarchy. Because these settings are defined at three different levels within Wwise, a sound, music, or motion object must pass through two separate processes before being played.

The first process occurs at the Actor-Mixer or Interactive Music level. When you define the advanced settings for objects at this level, you are setting a limit that is either global or per game object. If the limit is reached, the priority then determines which objects will be passed to the bus level in the Master-Mixer hierarchy.

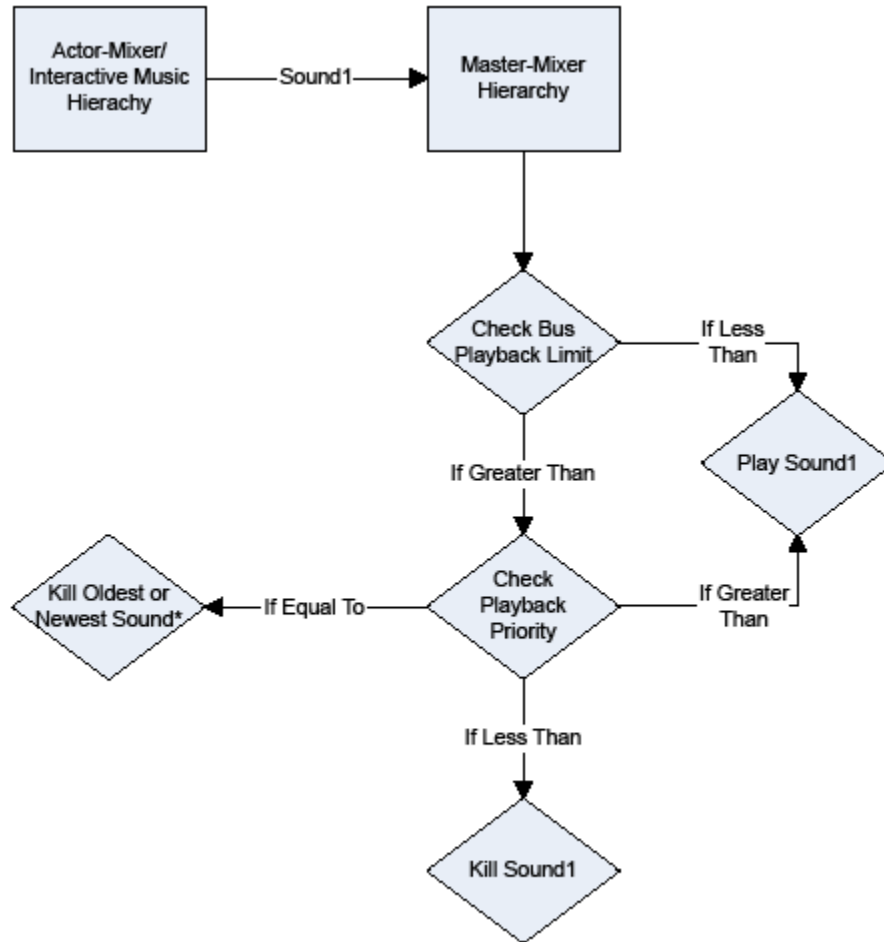
The following flowchart shows how Wwise determines which sounds and motion objects within the **Actor-Mixer Hierarchy** or which music objects within the **Interactive Music Hierarchy** will be played per game object:



* You decide whether oldest or newest sound is killed.

If the new sound, music, or motion object is not killed or sent to virtual voice at the Actor-Mixer or Interactive Music level, it passes to the second process at the Master-Mixer level. At this level, a global playback limit is used to restrict the total number of voices that can pass through the bus at any one time.

The following flowchart shows how Wwise determines which sounds, motion objects, and/or music will be output through a bus.



* You decide whether oldest or newest sound is killed.

At this point, the global project limit maximum number of voices determines if sounds are killed or put into virtual voice based on their own virtual voice settings.



Note

All the properties on the Advanced Settings tab of the **Property Editor** are absolute properties. For more information on absolute properties and overriding a parent's properties, refer to [Defining Absolute Properties](#).

Volume Threshold and Virtual Voices

Along with **Playback Limit** and **Playback Priority**, Wwise allows you to determine which sound, music, and motion objects will play based on a volume threshold. When the volume level reaches a specified volume threshold or when

the number of sounds is over the limit specified in the Playback Limit, the object can do one of the following:

- Continue to play.
- Be killed.
- Be moved into the virtual voice list.

The virtual voice list is a virtual environment where certain parameters of a sound or motion object are monitored by the sound engine, but where no processing occurs. If you select this option, sound, music, and motion objects move from the physical voice to the virtual voice and vice versa based on their volume levels. If the volume returns above the volume threshold or the number of playing sounds goes under the limit of simultaneous playing sounds, the objects automatically move back into the physical voice. For more information on setting the volume threshold, refer to [Specifying Volume Thresholds for a Project](#). For more information on defining the behavior of low-volume sounds and motion objects, refer to [Managing Low-Volume Sounds and Motion Objects](#).

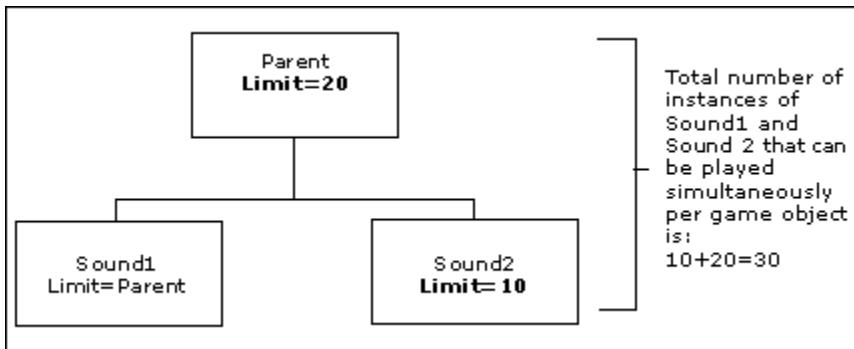
Limiting Object Playback Instances

To deal with limited game resources or game-design constraints, you must optimize the sounds, music, and motion objects that are playing at any one point in the game. You can manage the simultaneous playback of the various objects in your project using three different methods:

- Limit the number of sound, music, and/or motion instances that can be played per game object or globally on an **Actor-Mixer Hierarchy** object.
- Limit the overall number of sound, music, and/or music instances that can pass through a bus.
- Limit the overall number of sound, music, and/or music instances in the entire game.

When either limit is reached, Wwise uses the priority setting of the object to determine which one to discard based on the selected behavior to adopt when the limit is reached. If objects have equal priority, you have the option to discard the newest or oldest instance that is playing.

When you set a playback limit at the Actor-Mixer or Interactive Music level, you control the number of instances within the same structure that can be played either globally or per game object. If a child object overrides the playback limit set at the parent level, the total number of instances that can play is equal to the sum of all limits defined within the structure. This means that if, for example, you have a parent with a limit of 20 and a child with a limit of 10, the total possible number of instances is 30.



When you set the playback limit at the Master-Mixer level, it specifies the number of sound, music, and/or motion instances that can pass through the bus at any one time. Since the priority of each object has already been specified at the Actor-Mixer or Interactive Music level, there is no playback priority setting for busses.

When you set the playback limit at the project/game level, it specifies the number of sound, music, and/or motion instances that can be active at any one time. Each sound will adopt its own Virtual Voice behavior set in its Advanced Settings.

To define a playback limit:

1. Load a top-level parent object into the Property Editor.
2. Switch to the **Advanced Settings** tab.



Note

If the object is not a top-level object, you must select the **Override parent** option before you can set the **Playback Limit** options.

3. In the **Playback Limit** group box, select the **Limit sound instances to x** option.

The playback limit options become available.

4. In the corresponding text box, type the maximum number of instances that can be played simultaneously per game object.
5. To define the scope for the playback limit select one of the following options:

Global to apply the limit globally.

Per game object to apply the limit per game object.



Note

This option is not available on busses because the limit is always global.

6. To determine what happens when the playback is reached, select one of the following options:

Kill voice to stop the playing instance with the lowest priority.

Use virtual settings to adopt the defined virtual voice behavior for the sounds. The virtual sounds behaviors include: kill, send to virtual voice, and continue to play.

7. To determine what happens when the playback limit is reached and there is more than one sound, music, or motion object with the lowest priority, select one of the following options from the When limit is reached and priority is equal list:

Discard oldest instance to stop the oldest playing instance with the lowest priority.

Discard newest instance to stop the newest playing instance with the lowest priority.

8. The Project/Game limit can be modified in Wwise project settings, per platform. It may also be changed by the game using the Wwise SDK API once the sound engine is properly initialized.

When working with voice limitation system, it is good to know the following:

- Virtual voices do not count as valid voices.
- Sounds with the virtual voice setting set to “Continue to play” may cause the limit to be exceeded.

For example, let us suppose you have:

- A limit on bus set to 4 simultaneous sounds.
- The “Over limit behavior” on this bus is set to “Use virtual voice behavior”.
- The Volume threshold is set to -60 dB.
- You have 8 voices playing at this frame in this bus.

Sound name	Priority	Volume	Virtual Behavior
Sound_1	100	0 dB	Go Virtual
Sound_2	90	-90 dB	Go Virtual
Sound_3	80	-90 dB	Continue to play
Sound_4	70	0 dB	Go Virtual

Managing the Priority of Sounds and Motion

Sound name	Priority	Volume	Virtual Behavior
Sound_5	60	0 dB	Kill voice
Sound_6	50	0 dB	Kill voice
Sound_7	40	0 dB	Go Virtual
Sound_8	30	0 dB	Continue to play

The result would be:

Sound name	Result	Reason
Sound_1	Play (1/4)	Highest priority and over volume threshold.
Sound_2	Will go virtual	Volume under threshold.
Sound_3	Play (2/4)	Continues to play even when under volume threshold.
Sound_4	Play (3/4)	-
Sound_5	Play (4/4)	-
Sound_6	Will be killed	Over limit, already 4 sounds with higher priority are playing
Sound_7	Will go virtual	Over limit, already 4 sounds with higher priority are playing
Sound_8	Play (5/4)*	*Will play even if the limit is exceeded, this is a special situation where this sound is not allowed to go virtual nor be killed, and is not allowed to take over the 4 sounds already playing with a higher priority.

FAQ

Q: The limit does not seem to be working? Why?

A: Check if the sound's virtual behavior is not set to “Continue to play”, it is the default value for compatibility reasons. Any sound with this setting will not be kicked nor virtualized when over the limit (unless the over limit behavior was set to kill voice explicitly). Once you start using the “Use Virtual Voice settings” when over the limit, you must set this setting for any sound that would be considered “kickable” or “virtualizable” to start saving some CPU.

Q: The limit seems to be working most of the time but the profiler sometimes shows values that are over the specified limit even if none of these sounds are set to “Continue to play”.

A: When sounds are becoming virtual, they stay active for a few ms (one audio frame) to fade out and avoid clicking. For example, if the limit is set to 1 and one sound is already playing, then another sound pops in with a higher priority, the number of active voice will temporarily be to 2 while the first sound is fading out to go virtual.

Q: Why is sending to virtual voices not the default setting when over limit? Why are all sounds set to continue to play by default?

A: Some source codec may not be perfectly accurate in virtual mode and/or could require a seek table to be added in the source to be able to work in virtual voice: play from elapsed time. For this reason and to ensure that everything runs fine by default, this is not the default setting. For example, Vorbis will require a seek table to be able to properly adopt a virtual behavior with the From Elapsed time setting.

Related Topics

- [Defining Playback Priority](#)
- [Managing Low-Volume Sounds and Motion Objects](#)

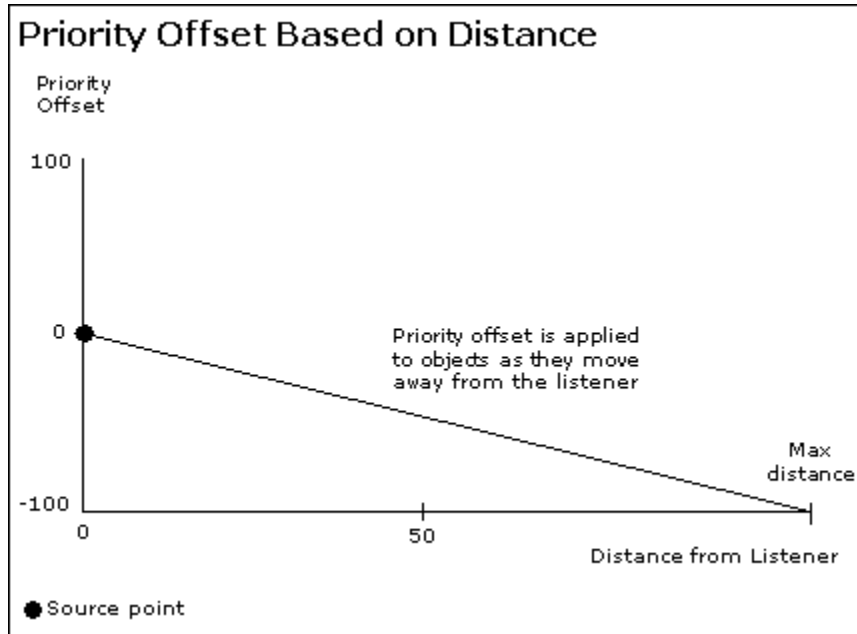
Defining Playback Priority

When you reach the limit of sounds, music, or motion objects that can be played at any one time, either at the game object or bus level, Wwise uses the priority or relative importance of each object to determine which ones will be played.

You can define the priority for each sound, music, or motion object using a standard numerical scale between 1-100, where 1 is the lowest priority and 100 is the highest priority. If objects have equal priority, you have the option to stop the newest or oldest instance that is playing.

Using Priority Offset Based on Distance

You can also alter the playback priority based on the distance the sound or motion object is from the listener. Wwise applies an offset to the priority using the Max distance values defined in the Attenuation Editor. The amount of offset applied will depend on the object's relative position to the listener. Wwise linearly interpolates the offset between the source point, where no offset is applied, and the attenuation max distance, where the full offset value specified is applied.

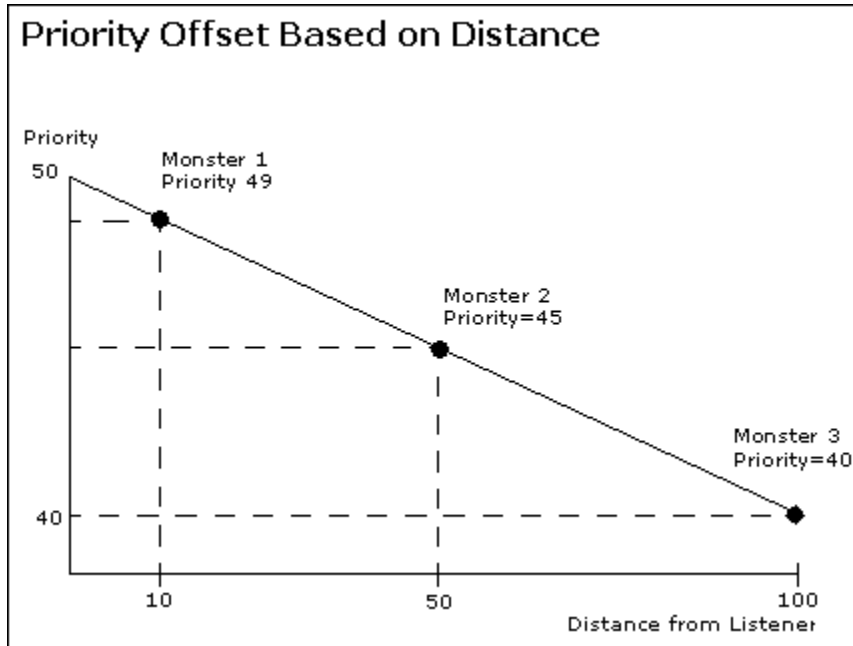


Using Priority Based on Distance - Example

Let's say you have three different sound objects: Monster1, Monster2, and Monster3. All three sound objects have a priority of 50 and a priority offset of -10. The Max distance in the Attenuation Editor is set to 100 meters. At any point in the game, these monsters will be at different distances from the listener. For the sake of this example, however, let's say that the monsters are located at the following distances from the listener:

- Monster 1: 10 meters
- Monster 2: 50 meters
- Monster 3: 100 meters

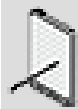
Because each monster is at a different distance from the listener and a priority offset based on distance is being applied, the priority for each sound will be different. The following graph shows how the distance from the listener will affect the priorities for each sound.



Since playback priority works together with playback limit, if the playback limit is surpassed, Wwise uses the priority of these sounds to determine which ones will play.

To define playback priority for sound, music, and motion objects:

1. Load a top-level parent object into the Property Editor.



Note

If the object is not a top-level parent object, you must select the **Override parent** option before you can set the Playback Priority options.

2. Switch to the **Advanced Settings** tab.
3. In the Priority text box, type a number from 1 to 100 that represents the priority or relative importance of the object.
4. To offset the playback priority based on the distance the object is from the listener, select the **Offset priority by x at max distance** option.
5. In the text box, specify the value by which you want the priority to be offset at the max distance.

Related Topics

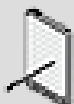
- [Limiting Object Playback Instances](#)
- [Managing Low-Volume Sounds and Motion Objects](#)

Managing Low-Volume Sounds and Motion Objects

When many sounds, music, and motion objects are playing simultaneously, you need to maintain an optimal level of performance. A good way to achieve this is to flag objects that fall below a certain volume level so that they don't take up valuable processing power and memory. Although you can continue to play these inaudible objects, you may want to either stop them outright or queue them in the virtual voice list.

The virtual voice list is a virtual environment where certain parameters of a sound or motion object are monitored by the sound engine, but where no processing occurs. The benefit of adding them to the virtual voice list is that these objects can move back and forth between the physical and the virtual voice based on their volume levels. As the volume falls below the volume threshold level, the sounds, music, or motion objects are added to the virtual voice list and processing stops. As volume levels increase, as is the case when objects move within the attenuation maximum distance radius, the sounds, music, and motion objects will move from the virtual voice to the physical voice where the audio or motion will be processed again by the sound engine. Despite the benefits of using virtual voices for low-volume sounds and motion objects, it is not the right solution for all objects due to certain limitations, including small delays and the lack of sample accuracy when returning to the physical voice.

To give you additional power and control, you can also define the playback behavior of objects as they move from the virtual voice back to the physical voice.



Note

You can set the volume threshold for your project in the Project Settings dialog box. For more information on setting the volume threshold levels, refer to [Specifying Volume Thresholds for a Project](#).

Managing Low-Volume Sounds - Example

Let's say you are creating a first-person shooter game where your main character is navigating through a series of hallways. He has picked up one of the enemy's two-way radios and is able to listen in on the enemy's communication. In this situation, you can have many sounds playing at the same time, including your character's footsteps, the torches burning in the dimly lit hallways, and the sounds coming from the two-way radio. Each of these sounds has different characteristics and requirements and therefore requires different treatments as their volumes change.

Sound	Treatment	Wwise Option
Footsteps	Footstep sounds are generally short, one-shot sounds, so when their volume falls below the volume threshold, you may want to kill these sounds to save valuable audio processing power.	Kill voice
Radio	Since the character within your game can turn the volume of the radio sounds up or down, you will need continuity, sample accurate precision, and immediate feedback for the radio communication. When the volume of these sounds falls below the volume threshold, you can continue playing these sounds. This enables you to generate an immediate and sample accurate response to when the game player turns the volume back up to continue listening to the enemy's communication.	Continue to play
Torch	Torch sounds are relatively short sounds that are looped continuously. Although these sounds require some continuity, they do not require the same amount of precision as the radio sounds. When the volume for these sounds falls below the volume threshold, you can send these sounds to the virtual voice list. The sound engine will monitor their volume levels but will not perform any audio processing until their volume levels return above the volume threshold.	Send to virtual voice

To manage low-volume sounds, music, and motion objects:

1. Load a top-level parent object into the Property Editor.



Note

If the object is not a top-level object, you must select the Override parent option before you can set the Volume Threshold options.

2. Switch to the **Advanced Settings** tab.
3. To specify the behavior of an object when it falls below the volume threshold, select one of the following options from the When below volume threshold list:

Continue to play to continue playing the sound, music, or motion object even though it will no longer be heard or felt. This is the only option that ensures sample accuracy.

Kill voice to stop playing the sound, music, or motion object.

Send to virtual voice to send the sound, music, or motion object to the virtual voice list, where certain parameters are monitored by the sound engine, but where no processing occurs.

4. If you selected the Set as virtual voice option, you must specify the behavior of the object when it moves from the virtual voice back to the physical voice. To do so, select one of the following options from the On return to physical voice list:

Play from beginning to play the object from its beginning. This option resets the object's loop count.

Play from elapsed time to continue playing the object as if it had never stopped playing.

Resume to pause the object when it moves from the physical voice to the virtual voice list and then resume playback when it moves back to the physical voice.



Note

Music objects returning to the physical voice will always use the Play from elapsed time option.

Related Topics

- [Limiting Object Playback Instances](#)
- [Defining Playback Priority](#)

Priority Tips and Best Practices

Before defining the advanced settings for your objects, you may want to review the following sections, which provide you with a series of tips and best practices that can help you get the results you want.

Playback Priority

Set music objects to higher priority - in most cases, the music objects in your game should have a higher priority than other sound or motion objects. This will ensure that your music plays back continuously in situations when many objects are triggered and the playback limit is surpassed.

Volume Threshold

Kill short sound FX when they fall below volume threshold - in most situations, you should use the Kill voice option for short sound FX that fall below the volume threshold because sending them to the virtual voice list will do either of the following:

- If **Restart from beginning** or **Resume** is selected, the sound will most likely be played outside its original context.

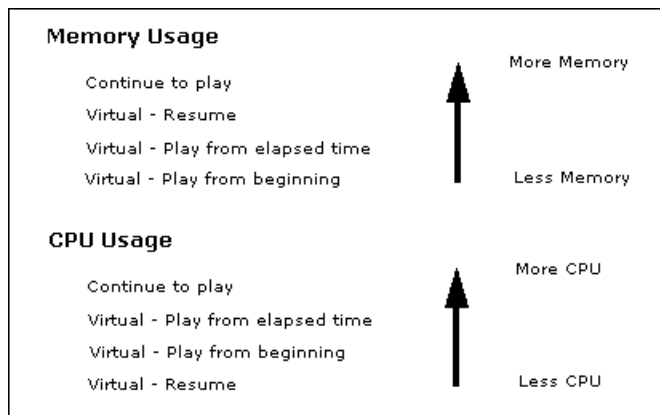
- If **Play from elapsed time** is selected, CPU will be used uselessly for a sound that will likely die as a virtual voice.

Virtual Voices

Before deciding which virtual voice option to use, you should be aware of the virtual voice settings' memory and CPU usage:

- **Restart from beginning** - This option uses a small amount of memory and CPU, but if the sound is streamed, there may be a delay when it returns from the virtual voice.
- **Resume** - This option uses a smaller amount of CPU but a larger amount of memory because it keeps the memory buffers on hold for when the sound returns from the virtual voice. Be aware that if a sound never returns from the virtual voice because it never re-enters the radius, the memory buffers will remain in memory for the length of the game. If this occurs for many sounds, the memory buffers will accumulate over time and may end up taking a significant amount of memory in the game.
- **Play from elapsed time** - This option can save some CPU and memory, but if the sound is streamed, there may be a delay when it returns from the virtual voice.

The following illustration displays the memory and CPU usage for each of the virtual voice options in relation to each other and to the **Continue to Play** option:



Of course, when you take into account a sound's audio format, sample rate, streaming settings, and so on, the differences between these options can go from negligible to very large.

Always Calculated	Not Calculated When Virtual
<ul style="list-style-type: none"> • Positioning, calculations, voice priority, and volumes are still computed because they are required to determine if the voice should return physically or not. 	<ul style="list-style-type: none"> • Codec decompression. • Audio data moving, copying, or mixing.

Always Calculated	Not Calculated When Virtual
<ul style="list-style-type: none">• In from elapsed time mode, source plug-ins are still computed when virtual (because the sound engine cannot seek in a source plug-in).	<ul style="list-style-type: none">• LPF, sample-rate conversion, pitch shifting, and effects inserted in the Actor-Mixer Hierarchy.• I/O reads.



Note

Most of the time, buffers of audio data are freed when the voice becomes virtual, except for the **Resume** behavior.



Note

Effects on busses of the **Master-Mixer Hierarchy** are processed only when there is at least one physical voice playing through this bus.

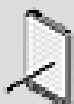
Playback Limits on the Wii

The maximum number of voices on the Wii is determined by the hardware and will vary depending on the Audio DSP usage. Voices that cannot play due to hardware limitations will use the volume threshold settings of the related sound to determine what will happen to them. This means that some voices on the Wii may be sent to the virtual voice list even if their volume does not fall below the threshold level.

For example, let's say the maximum number of hardware voices is 96 and at a particular point in the game 100 voices have been triggered to play. In this case, the 4 voices with the lowest priority will do one of the following based on each corresponding sound's volume threshold setting:

- If set to “Continue to play”, the voice will be sent to the virtual voice list.
- If set to “Kill voice”, the voice will be killed.
- If set to “Send to virtual voice”, the voice is sent to the virtual voice list.

When a voice is sent to the virtual voice list on the Wii, its volume gets computed twice less often to check if the volume must return from the virtual state. Although this saves a significant amount of CPU for voices that are not loud enough to be heard, virtual voices on the Wii may be delayed by a few milliseconds when returning from the virtual voice.



Note

Stereo voices take two hardware voices each. If a stereo voice gets one of its two voices killed or sent to the virtual voice list because

too many voices are playing, both voices of a stereo sound will follow the same behavior to ensure that they have consistent positioning.

Using Instance Limiting with Music Objects

When using instance limiting with music objects, be careful not to set the limit too low because it may prevent certain clips from playing causing breaks or cuts in your music. Remember that when fading and switching between music segments, you can easily use up several instances, with the different pre-entry and post-exit parts of one or more clips within each segment, as well as any stinger segments that might be playing.

Chapter 12. Managing Effects

Overview	330
Using Effects	330
Using Effects to Implement Environment Acoustics	340
Effects Tips and Best Practices	344

Overview

One way to enhance the sounds, music, and motion within your game is to apply one or more effects to the objects in your project's hierarchy. Using Wwise's open architecture, you can easily create and define your own effect plug-ins. These varied effects can add more vibrancy and realism to your game's sounds, music, and motion. In addition, to assist you in this process, Wwise comes with several effect plug-ins and factory sharesets.

For more information about rendering effects, refer to [Rendering Effects](#). For more information about effects used to recreate environment acoustics, refer to [Using Effects to Implement Environment Acoustics](#).

Using Effects

Effects are made up of a collection of properties called an instance. When you define an instance, you can use it as a ShareSet, which allows you to apply it to many different objects within your project. The advantage of using a ShareSet is that you don't have to modify the effect's properties for each object individually. However, if you prefer, you can also define custom instances tailored for one specific object.

When changes are made to the effect properties of a ShareSet, all objects subscribing to that ShareSet are affected.

You can carry out the following tasks to help manage effects in your project:

- [Creating Effect ShareSets](#)
- [Deleting Effect ShareSets](#)
- [Applying Effects to Objects](#)
- [Applying Effects to Busses](#)

Creating Effect ShareSets

Most often, when you apply an effect to an object or bus in the hierarchy, you apply this effect as a ShareSet. Wwise comes with several default and pre-defined effect ShareSets, but you will probably want to expand this list by creating your own. Each effect ShareSet provides you with a different version of an effect that you can then apply to various objects in your project.

You can create effect ShareSets in two locations in Wwise:

- In the Project Explorer
- In the Property Editor

To create an effect ShareSet in the Project Explorer:

1. In the Project Explorer, switch to the ShareSets tab.

2. In the Effects section, right-click the work unit where you want to create the ShareSet.

The shortcut menu is displayed.

3. Click **New Child**.

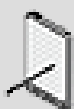
The list of available effect plug-ins is displayed.

4. Select the type of effect you want to create.

A new instance appears as a child of the work unit in the hierarchy.

5. Replace the default name with one that best represents the new ShareSet and press **Enter**.

The new name of the ShareSet, along with the type of effect in brackets, is displayed in the Effects hierarchy.

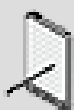


Note

Each ShareSet name must be unique, and must only include letters, numbers, and underscores. It must begin with a letter or number. You can rename a ShareSet at any time by right-clicking it, selecting **Rename**, and typing a new name. Unless absolutely necessary, you shouldn't rename a ShareSet after it has been integrated into the game because it will require additional programming.

To create an effect ShareSet from within the Property Editor:

1. Load an object into the Property Editor.
2. Switch to the Effects tab.



Note

If the object is not a top-level object, you must select the **Override parent** option before you can set the Effect options.



3. In the Effects table, click the Selector button (>>).

The list of available effects is displayed.

4. Place the mouse cursor over the type of effect you want to apply.

A list of corresponding effect ShareSets is displayed.

5. Click **New**.

The **New Effect** dialog box opens.

6. Select the work unit in which you want to create the effect ShareSet.
7. Type a name for the ShareSet.



Note

Each ShareSet name must be unique, and must only include letters, numbers, and underscores. It must begin with a letter or number. You can rename an instance at any time by right-clicking it, selecting **Rename**, and typing a new name.

8. Click **OK**.

The new ShareSet is created, and is applied to the selected object.

Related Topics

- [Copying Settings from One Platform to Another](#)
- [Deleting Effect ShareSets](#)
- [Applying Effects to Objects](#)
- [Converting Effect ShareSets into Custom Instances](#)

Deleting Effect ShareSets

You can delete a ShareSet if you no longer need it. Before deleting a ShareSet, you should make sure that objects are not still using it. If you delete a ShareSet, the ShareSet is automatically removed from all objects that subscribe to it.

To delete a ShareSet:

1. In the Project Explorer, switch to the **ShareSets** tab.
2. In the Effects section, click the ShareSet you want to delete.
3. Press the **Delete** key.

The ShareSet is deleted, and is removed from any objects to which it was applied.

Related Topics

- [Creating Effect ShareSets](#)
- [Applying Effects to Objects](#)

Applying Effects to Objects

Effects can add realism and variety to the sounds, music, and motion objects in your project. After you have created your effect ShareSets, you can easily edit them and apply them to your project's sounds and motion objects. You can also convert an effect ShareSet into a custom effect instance, depending on whether you want its properties to affect one object or several.

You can apply up to four different effects to each object within your hierarchy. If a chain of effects is applied, the effects are processed in the order in which they appear in the list. The order in which effects are applied is important as it alters the end result of your sound or motion object. Some effects can also be quite CPU intensive, so it is important that you develop an efficient strategy for using them. For some tips on applying effects, refer to [Effects Tips and Best Practices](#).

When applying effect instances, you may want to do the following:

- [Applying an Effect ShareSet to an Object](#)
- [Converting Effect ShareSets into Custom Instances](#)
- [Bypassing an Effect](#)
- [Rendering Effects](#)
- [Editing Effect Properties](#)
- [Re-Ordering Effects](#)

Applying an Effect ShareSet to an Object

After a ShareSet has been created, you can apply it to any object within your project hierarchy.



Tip

Applying time-based effects (such as Wwise RoomVerb or Delay) to music objects is not recommended, as this can interfere with time-based properties and behaviors already assigned to these objects. To avoid this interference, you can instead apply the effect at the master-mixer level. For more information, refer to [Applying Effects to Busses](#).

To apply an effect ShareSet to an object:

1. Load an object into the Property Editor.
2. Switch to the Effects tab.



Note

If the object is not a top-level object, you must select the Override parent option before you can set the Effect options.



3. In the Effects table, click the Selector button (>>).

A list of effects that can be applied to the object is displayed.

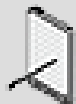
4. Place the mouse cursor over the type of effect you want to apply.

A list of corresponding effect ShareSets is displayed.

5. Click the ShareSet you want to apply.

The ShareSet is applied to the object.

6. Repeat steps 3-5 to apply additional effects to the object.



Note

You can apply up to 4 different effects to an object.

Related Topics

- [Editing Effect Properties](#)
- [Converting Effect ShareSets into Custom Instances](#)
- [Bypassing an Effect](#)
- [Rendering Effects](#)
- [Editing Effect Properties](#)
- [Re-Ordering Effects](#)

Converting Effect ShareSets into Custom Instances

Effect instances in Wwise fall into two categories:

- **Custom effect instances** that are applied to a single sound, music, or motion object only. When you change the properties of a custom instance, only this object is affected.
- **ShareSets** that can be applied to many objects. When you change the properties of a ShareSet, all the objects using that ShareSet are affected.

By default, all effect instances you create begin as ShareSets. However, you might want to tweak a ShareSet for only one object. To do this, you can convert

the effect ShareSet into a custom effect instance. For example, let's say you've created a ShareSet called Funky_Reverb, but you want a special variant of that ShareSet just for a sound called Speech_1. You can apply Funky_Reverb to Speech_1, convert it to a custom instance, and then make your changes. The changes you make will apply only to Speech_1, and not to the objects that subscribe to the Funky_Reverb ShareSet.

To convert a ShareSet into a custom instance:

1. Load an object which has the effect applied to it into the Property Editor.
2. Switch to the **Effects** tab.
3. In the Effects table, select **Define custom** from the Mode list for the effect that you want to convert to a custom instance.

The name of the effect instance has the word “Custom” appended to it. From now on, changes you make to the instance will only affect the one object that uses it.



Note

You can also create a custom effect instance by applying the Default (Custom) version of a ShareSet to an object. The original ShareSet still exists, and remains unchanged for any subscribed objects.

Related Topics

- [Applying an Effect ShareSet to an Object](#)
- [Editing Effect Properties](#)

Bypassing an Effect

After you've applied an effect to an object, you can always audition the original unprocessed version by bypassing the effect. Since you can apply several effects to an object, you can bypass each one individually. You can also bypass an effect at any level in the hierarchy.



Note

You can also bypass an effect in-game by using the Bypass action in the Event Manager, or by using an RTPC.

To bypass an effect:

1. Load a top-level object into the Property Editor.

2. Switch to the **Effects** tab.

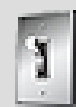


Note

If the object is not a top-level object, you must select the **Override parent** option before you can set the Effect options.

3. In the Effects table, select the **Bypass** option for each effect that you want to bypass.

The effect(s) are no longer processed.



Tip

To bypass all the effects applied to your object, select the **Bypass All** option.

Related Topics

- [Editing Effect Properties](#)
- [Converting Effect ShareSets into Custom Instances](#)
- [Rendering Effects](#)
- [Editing Effect Properties](#)
- [Re-Ordering Effects](#)

Rendering Effects

You can render one or more effects that have been applied to sound, music, or motion objects prior to packaging them into SoundBanks. This saves on processor power during gameplay, but prevents you from bypassing the effect using the **Bypass FX** event or from using RTPC curves on these effects.

When you render an effect within an effect chain, all previous effects will automatically be rendered as well. For example, let's say you have a sound object with the following three effects applied to it:

- Matrix Reverb
- Parametric EQ
- Compressor

If you select the render option for the Parametric EQ effect, both the Parametric EQ and Matrix Reverb effects will be rendered prior to packaging into SoundBanks.



Note

All the Wwise effect plug-ins, including Delay and SoundSeed Impact, must be rendered when using on the Wii platform. When you apply effect instances to objects that are linked on multiple platforms, you must render these effects so that they can be used on the Wii. Refer to the [Overview](#) for a complete list of effects that need to be rendered on each of platform.

To render an effect:

1. Load an object into the Property Editor.
2. Switch to the **Effects** tab.



Note

If the object is not a top-level object, you must select the **Override parent** option before you can set the Effect options.

3. In the Effects table, select the **Render** option for each effect that you want to render.

The selected effect(s) as well as all previous effects will be pre-rendered on the object when the SoundBanks are generated.



Note

For more information on generating SoundBanks, refer to [Generating SoundBanks for a Project](#).

Related Topics

- [Editing Effect Properties](#)
- [Converting Effect ShareSets into Custom Instances](#)
- [Bypassing an Effect](#)
- [Editing Effect Properties](#)
- [Re-Ordering Effects](#)

Editing Effect Properties

You can edit the properties of any effect instance to suit your needs. In particular, you will probably want to edit the properties of new ShareSets or

custom instances before applying them. If you don't, the effect properties will be left at their default settings. You can further enhance the effect instance by assigning RPTCs to some of its properties.

When you change the property values of a ShareSet, all objects using that instance will be affected. When you edit a custom effect instance, only the object to which the effect is applied will change.



Note

You can use presets to store effect properties and speed up your creation of multiple effect instances. For more information on presets, refer to [Using Presets](#).

To edit a ShareSet:

1. In the ShareSets tab of the Project Explorer, double-click the ShareSet you want to edit.

The Effect Editor for the selected effect opens.

2. Define the effect properties as you want them to sound in your project.



Note

For a complete description of the effect properties, click the Help button in the corresponding Effect Editor.

To edit a custom effect instance:

1. Load an object or bus into the Property Editor.
2. Switch to the **Effects** tab.
3. In the Effects table, click the Edit icon (...) for the custom instance you want to edit.

The Effect Editor for the selected effect opens.

4. Define the effect properties as you want them to sound in your project.



Note

For a complete description of the effect properties, click the Help button in the corresponding Effect Editor.

5. To return the values of the properties in your custom effect instance to those of the ShareSet on which it is based, click **Reset Effect Settings**.

Related Topics

- [Converting Effect ShareSets into Custom Instances](#)
- [Applying an Effect ShareSet to an Object](#)
- [Bypassing an Effect](#)
- [Rendering Effects](#)
- [Re-Ordering Effects](#)

Re-Ordering Effects

The order in which effects are processed can have a dramatic impact on the end result of your sound or motion object. Therefore, it is important that you apply effects in the correct order. Wwise processes the effects in the order in which they appear in the table. If you need to, you can easily re-order the list of effects by dragging an effect to a different position in the list.

To re-order effects:

1. Load an object or bus into the Property Editor.
2. Switch to the **Effects** tab.
3. In the Effects table, select the effect that you want to move up or down in the list.

The entire row becomes highlighted.

4. Drag the effect to the new position in the list.

A red line indicates where the effect will be placed in the list.

Related Topics

- [Converting Effect ShareSets into Custom Instances](#)
- [Applying an Effect ShareSet to an Object](#)
- [Bypassing an Effect](#)
- [Rendering Effects](#)
- [Editing Effect Properties](#)

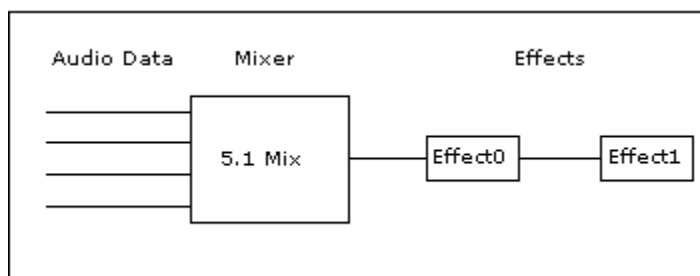
Applying Effects to Busses

You can apply audio effects to busses as easily as applying them to sound, music, or motion objects. All you have to do is load the bus into the Property Editor and apply the effects as you would to a sound object. For more information on this process, refer to [Applying Effects to Objects](#).

There are some limitations when applying effects to a bus, including:

- Effects are currently not supported for motion busses.
- The SoundSeed Impact plug-in can't be applied to busses.

When effects are applied to a bus, all incoming audio data is sub-mixed before the effect is applied. If a chain of effects is applied, the effects are applied in the order in which they appear in the list.



Note

The Wii doesn't support real-time effects. When you are working on the Wii platform, you can use the option **Enable Wii Compressor** in the Audio Bus Property Editor to apply a compressor to the Master Audio Bus.

Using Effects to Implement Environment Acoustics

In Wwise, there are two different types of bus: audio busses and auxiliary busses. The auxiliary busses, along with the send system can be used to implement environment acoustics. Effects applied on auxiliary busses can be used to re-create how the environment influences the playing sounds, for example by adding reverberation. Effects on auxiliary busses have the additional capability of being applied dynamically according to game object location data.



Note

Effects can't be applied to motion busses in the Master Motion Bus hierarchy.

The following sections will show you how you can use effects in auxiliary busses in your project:

- [Understanding Sends](#)
- [Integrating Game-defined Auxiliary Sends Into Your Workflow](#)

Understanding Sends

The acoustics of an environment within your game is simulated by adding a series of effects in an auxiliary bus and sending the signal to that bus. You can create as many auxiliary busses as you want; one for each of the different environments in your game. You also have the possibility to add effects to an auxiliary bus dynamically in the game using the SDK function `SetBusEffect()`. Once you have defined the auxiliary busses with your effects, you need to define which object is sending to these busses.

You have the choice of defining the object sends statically by using user-defined auxiliary sends, or by letting your game define the sends by using the game-defined auxiliary sends.

To create an auxiliary bus:

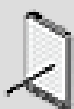
1. In the Master-Mixer hierarchy, create an auxiliary bus object
2. Inspect the auxiliary bus
3. In the effect tab of the auxiliary bus, add the required effects for the environment acoustics

To have an object affected by an environment acoustics using the user-defined auxiliary sends:

1. Inspect the Actor-Mixer Hierarchy object
2. Drag & Drop an auxiliary bus to the User-defined auxiliary sends list
3. Use the Send volume slider to define how much signal is sent to the auxiliary bus, defining how much the object is participating in the environment acoustics.

To have an object affected by an environment acoustics using the game-defined auxiliary sends:

1. Inspect the Actor-Mixer Hierarchy object
2. Enable the option *Use game-define auxiliary sends*
3. In the game, use the function `AK::SoundEngine::SetGameObjectAuxSendValues()` to assign an auxiliary send per game object.



Note

For more information on mapping environments in-game, refer to [Wwise SDK - Windows > Sound Engine Integration Walkthrough > Integrate Wwise Elements into Your Game > Integrating Environments and Game-defined Auxiliary Sends in](#)

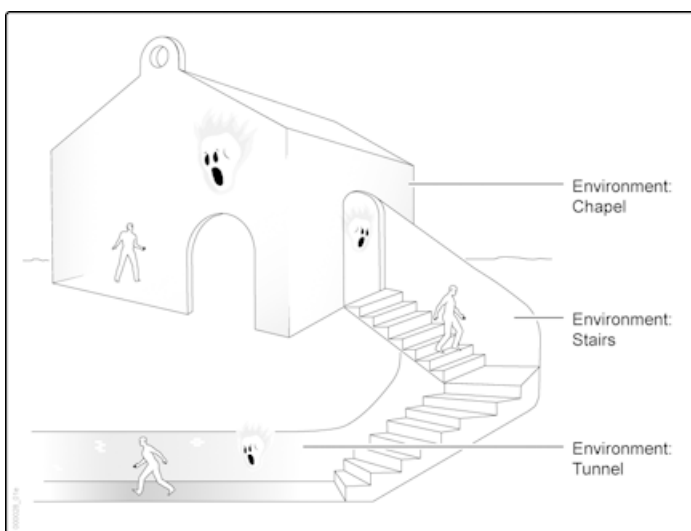
the [Wwise SDK documentation](#). You and the audio programmer can also look at the Integration Demo which shows you examples of how to setup auxiliary sends.

Using Auxiliary Sends to Recreate Environment Acoustics - Example

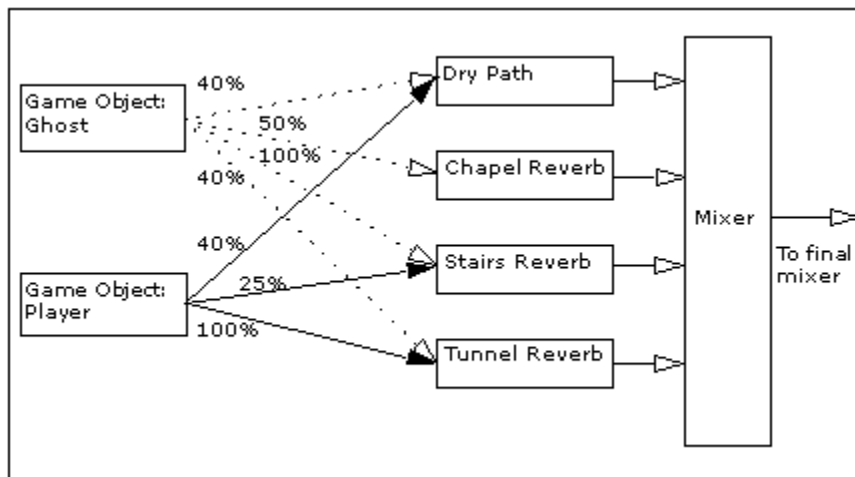
Let's say you are working on a game that takes place in and around a haunted graveyard. The game is full of ghosts, and you want the ghosts to sound different depending on which environment the ghost sounds are coming from. In one section of the game, your player can explore a chapel, a tunnel, and the stairway connecting the two. For this section of the game, you define the following auxiliary busses:

- Chapel
- Stairs
- Tunnel

You decide to give each of these three environments a distinct Reverb. For example, the tunnel is a much smaller space than the chapel, and has cavernous stone walls; therefore, its reverb will be much more pronounced than that of the chapel. In Wwise, you can create a reverb instance that will have a higher reverb level and shorter decay time to match the acoustics in the tunnel, as well as others to match the chapel and the stairs. Later, a developer maps the auxiliary busses that you have created to locations in the game's geometry. As a result, when a ghost is in the tunnel, ghost sounds echo far more than when the ghost is in the chapel.



Wwise auxiliary sends can also emulate the movement between environments by dynamically calculating the send volume for all of the simultaneous sends. The following diagram illustrates the in-game dynamic auxiliary bus routing and mixing for this example.

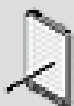


The proportion of each send that is applied to the particular sounds depends on the position of each game object within the game geometry. Here, your intrepid player is descending the stairs from the chapel into the tunnel, with a ghost in close pursuit. Partway through the tunnel, the player and ghost can be defined as being 100% in the Stairs environment, but also 50% in the Chapel environment, and 40% in the Tunnel environment. The ghost's sounds are then processed with each reverb instance at the appropriate percentage.

Integrating Game-defined Auxiliary Sends Into Your Workflow

In Wwise, you can create and edit auxiliary busses with effects and define which object is sending to them. Auxiliary sends can be defined programmatically to sounds based on game object positioning. The following steps describe how a development team can implement game-define auxiliary sends:

- The sound designer defines auxiliary bus objects corresponding to environments in the game, such as small room, church, and cave.
- The sound designer makes sure all sounds that are to be affected by the environment acoustics have the option *Use game-defined auxiliary sends* enabled.
- The game developer maps the auxiliary busses to the environments as they appear in the game geometry.
- The game developer creates a mechanism for reporting the real-time calculation of percentages of environments to the sound engine.
- The sound engine calculates which auxiliary bus apply to the sounds triggered by each game object.



Note

To audition and tweak game-defined auxiliary sends, you must either connect to the game and tweak your effect in real time,

or apply the auxiliary send as a user-defined send temporarily. You may also want to try the SoundFrame Test tool (SFTTest.exe), which is installed as part of the Wwise SDK. This tool allows you to remote-control Wwise and retrieve data from the project. You must attach an event to a game object, play it, and then edit the game-defined auxiliary sends by pressing the Edit Sends button. This will allow you to control the various levels of your game-defined sends.

Effects Tips and Best Practices

Many options are available to you when you use effects in Wwise. Using certain strategies when implementing effects can give you better-sounding results and save resources. The following are some strategies you might consider when using effects in your projects.

CPU Usage

Effects will always use up CPU power, but being aware of this power draw can help you use them more efficiently. In general, if you apply effects at the Master Mixer level, you will use less CPU. For example, if you apply a Wwise Peak Limiter effect on the Master Audio Bus of an Xbox 360 game, you will have only one instance processed at runtime. If you applied the effect at the object level instead, you could have hundreds of instances processed at once.

Rendering effects also allows you to save on CPU, as this will avoid the need to process them in real time. Naturally, you cannot apply RTPCs to rendered effects as the properties of these effects cannot change after they have been rendered.

As far as individual effects go, Delay and Wwise Parametric EQ tend to use very little CPU. The Wwise Compressor, Peak Limiter, and Expander effects use somewhat more. For reverb, you have a choice between the Wwise RoomVerb, which is more resource-intensive but of high quality; and the Wwise Matrix Reverb, which you can adjust to suit your quality and performance needs.

Overall, your best strategy is to test your project using the game profiling tools. In this way, you can observe the CPU usage of effects in real time and make decisions about how effects should be used. For more information about profiling, refer to [Understanding the Different Types of Profiling in Wwise](#).

Effects and Music

Applying time-based effects (such as Wwise Matrix Reverb or Delay) to music objects at the object level is not recommended, as this can interfere with time-based properties and behaviors already assigned to them. Applying the time-

based effects at the master-mixer level, that is, to an audio bus, avoids this interference.

Chapter 13. Managing Motion

Overview	347
Understanding How Motion Works in Wwise	347
Creating Motion for Your Game	350
Building an Output Structure for Motion	358
Motion Tips and Best Practices	359

Overview

Another great way to further immerse the player in a game, is to add some kind of motion feedback.

Wwise offers a complete pipeline for creating and integrating motion in your game. By implementing a comprehensive pipeline solution for motion, similar to the one that exists for building audio, Wwise allows you to:

- Create sophisticated and realistic motion effects with a very short learning curve.
- Integrate motion easily into a game without significantly affecting the performance of the game or sound engine.
- Use the same features as audio to build and integrate motion.
- Create motion effects for the same type of device on various platforms without additional work.
- Add or remove the motion component easily based on the requirements of your game.



Note

Wwise Motion is not available on the Mac® or on iOS platforms.

Understanding How Motion Works in Wwise

In an attempt to make the transition from audio to motion as simple as possible, Wwise uses the same workflow and many of the same principles and features for both. Like audio, motion can be structured within a hierarchy, output to a bus, and triggered in game using events. It can also be positioned within a 3D environment, manipulated by the various game syncs in your project, and packaged into SoundBanks.

For a comparison between the features available for audio development to those available for motion development, refer to the following table:

Wwise Feature	Audio	Motion
Bus routing	✓	✓ Master Motion Bus hierarchy
Object Properties and Behaviors	✓	No LPF control
Object Hierarchy	✓	✓
Effects	✓	✓ No effects on motion busses

Wwise Feature	Audio	Motion
Positioning	✓	Limited 2D/3D support for controllers
Events	✓	✓
RTPCs	✓	✓
States	✓	✓ No LPF control
Switches	✓	✓
Playback Priority	✓	✓
Playback	✓	✓ Motion device required
Platform Customization	✓	✓
Simulations	✓	✓
Profiling	✓	✓
SoundBanks	✓	✓
Queries	✓	✓

Types of Motion Devices

Currently, Wwise supports the following types of motion devices:

- Game controllers (all platforms and generic types)

PS3™ and PS4™ Move

Wii® Remote

All devices have been grouped together to facilitate multi-platform development. You must enable "Game controllers" in the Project Settings, before you can create and integrate motion in your project. For more information about enabling motion devices, refer to [Enabling the Motion Devices for Your Project](#).

Methods for Generating Motion

At a very basic level, motion data is generated in Wwise from a source. This source can either be an existing media file, a new media file, or a plug-in signal generator.

After you have decided which motion devices will be supported by your game, you must then decide which method you will use to generate motion. In Wwise, there are two different methods for creating a motion source:

- Using an existing audio signal.
- Using a motion FX object.

Generating a Motion Source from an Existing Audio Signal

When you convert an existing audio signal into a motion source, the audio signal is split in two at run-time after both RTPCs and effects have been applied. The split is done so as not to affect the original sound. Since audio has a much larger spectrum than motion, the higher frequencies are filtered out using a low pass filter. The signal is then re-sampled using a much lower sample rate to create the motion source.



Note

The LFE channel is ignored when generating motion from an existing audio source.

Since the motion source is generated from an existing audio source, the motion is tied to the audio playback in game. This means that the motion source does not require a separate event to be triggered in game. It also means that the motion source is affected by the same properties, behaviors, game syncs, and so on, as the audio object.

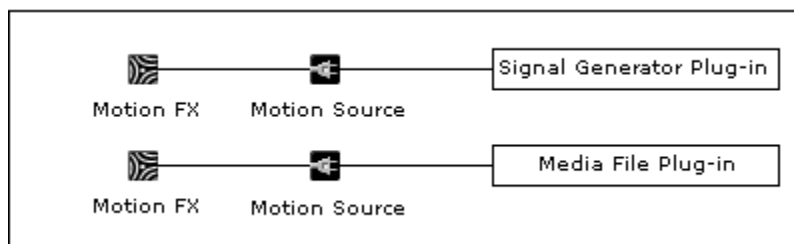


Note

You can't generate a motion source from an existing audio source on the Wii platform. To generate motion on the Wii, you have to use motion FX objects and the motion generator plug-in.

Generating Motion Using a Motion FX Object

Another way to generate motion is by creating special Wwise objects, called Motion FX objects. These objects, like sound objects, contain a source. The motion source can be created from a plug-in signal generator.



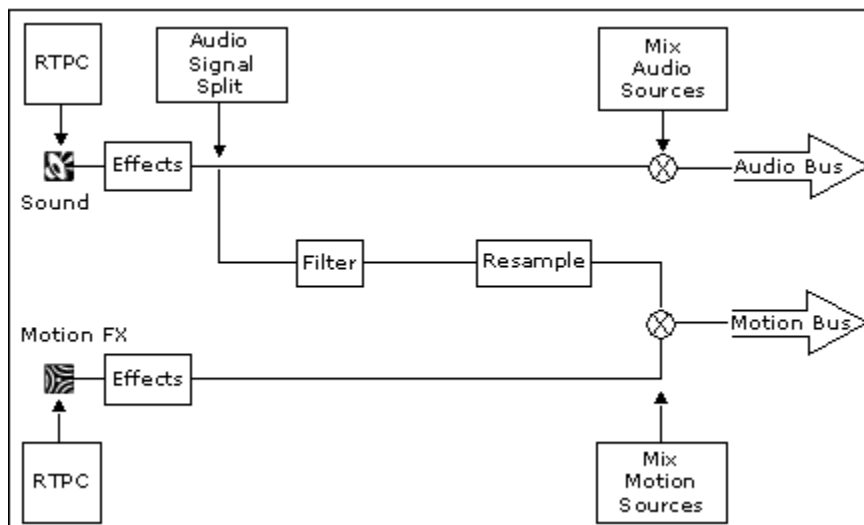
When using Motion FX objects, you can build sophisticated motion structures, using containers and actor-mixers, to define the properties and behaviors of

your motion effects. Since these motion effects are not necessarily tied to the audio in your game, they can be triggered at any point in game by their own events.

Of course, there will be situations where one method will be preferential over another. For a further discussion on the benefits of each of these methods, refer to [Creating Motion for Your Game](#).

Motion Processing Pipeline

The motion processing pipeline takes the motion signal generated from both types of sources and mixes them together before passing this mix through a special motion bus. The following illustration demonstrates how Wwise processes both types of motion sources and how the motion pipeline fits into the audio processing pipeline.



Creating Motion for Your Game

In Wwise, you can generate motion using either of the following two methods:

- [Generating Motion from Existing Sounds](#)
- [Using Motion FX Objects to Generate Motion](#)

The requirements and limitations of each situation will determine which method you choose. Take the time to understand each method's strengths and weaknesses so that you can use them appropriately.



Note

Before you can create motion for your game, you need to specify which motion devices you will support in your project. For more

information on enabling motion devices, refer to [Enabling the Motion Devices for Your Project](#).

Generating Motion from Existing Sounds

Since many motion effects will be linked to audio events, such as gun shots, road vibrations, explosions, and so on, you can generate motion data directly from the existing sounds and music objects in your project. In this case, Wwise simply takes the audio signal, extracts the low-frequency components and then re-samples this part of the signal to give a convincing motion effect. Although this method requires very little effort on the part of the designer and developer and is perfect for motion that is synced with audio, it does have the following disadvantages:

- Less control over the type of motion that is created.
- More CPU intensive as the audio signal must be filtered and re-sampled at run-time to create the motion data.
- Does not handle cases where motion is not related to audio.

Despite these disadvantages, there are situations where this method can be useful. For example, this method is perfect for creating prototypes of the motion in your game. When the prototypes are completed and approved, you can then decide whether to go on to create specific motion FX objects that use less CPU and allow for more detail.

Generating motion from existing sounds involves the following tasks:

- [Specifying Output Routing for Motion](#)
- [Boosting/Reducing the Motion Signal](#)
- [Specifying the Motion Signal's Cutoff Frequency](#)

Before generating motion from existing sounds, you should be aware of the following:

- The LFE channel in the original audio source is ignored.
- The removal of the DC offset from the original audio source will alter the motion output.
- The motion output will feel different on each platform's controller due to differences in the motors' power, top speed, and weight.
- A motion source can't be generated from an existing audio source on the Wii platform. To generate motion on the Wii, you have to use motion FX objects and the motion generator plug-in.

Specifying Output Routing for Motion

If you plan to generate motion data from an existing audio source, you need to first specify through which motion bus the motion data will be routed. Motion

data is different than audio data, and requires a separate hierarchy of motion busses through which the motion data can be routed. You can create this separate hierarchy under the Master Motion Bus in the Master-Mixer hierarchy. For more information on creating a motion bus hierarchy, refer to [Overview](#).

To specify the output routing for motion data:

1. Load a top-level object into the Property Editor.



Note

If the object is not a top-level object, you must select the Override parent option before you can set the Motion Output options.

2. In the Motion Output Bus group box, click the **Browse** button (...).

The Project Explorer - Browser opens.

3. Select the motion bus through which you want the motion data to be routed.
4. Click **OK**.

The motion data generated from the current object and any child objects below it is now routed through the selected motion bus.

Related Topics

- [Boosting/Reducing the Motion Signal](#)
- [Specifying the Motion Signal's Cutoff Frequency](#)

Boosting/Reducing the Motion Signal

When Wwise generates motion data from an existing sound or music object, it filters out the high frequencies of the audio signal and then uses the low frequencies to create the motion signal. This generated motion signal may not have the intensity that you are looking for, so you can amplify or attenuate the signal using the Motion Volume Offset property. This will apply an offset to the sound's volume setting, which will alter the intensity of the motion effect.

To boost or reduce the generated motion signal:

1. Load an object into the Property Editor.
2. In the Audio to Motion Settings group box, modify the Motion Volume Offset value to either amplify or attenuate the motion signal.



Note

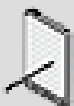
To test the intensity of a motion effect in Wwise, you must connect a motion device to your workstation and then play back the Wwise object.

Related Topics

- [Specifying Output Routing for Motion](#)
- [Specifying the Motion Signal's Cutoff Frequency](#)

Specifying the Motion Signal's Cutoff Frequency

Wwise uses the low frequencies of an audio signal to generate the corresponding motion signal. To give you additional control and flexibility over which frequencies are kept and which ones are discarded, Wwise allows you to set the cut off frequency using the Motion Low Pass property.



Note

The Motion Low Pass property only determines which frequencies are extracted to generate the motion signal. It has no impact on the original sound.

To specify the motion signal's cut-off frequency:

1. Load an object into the Property Editor.
2. In the Audio to Motion Settings group box, modify the Motion Low Pass value to set the cutoff frequency that will be used to generate the motion signal.

Related Topics

- [Specifying Output Routing for Motion](#)
- [Boosting/Reducing the Motion Signal](#)

Using Motion FX Objects to Generate Motion

Due to the limitations of generating motion from existing audio sources, you may want to use separate Wwise objects to create and integrate the motion assets in your project. By creating motion FX objects, you have more flexibility and control over what type of motion is created. Since the motion data is not converted from existing audio sources, you can generate motion using far fewer

samples, in the range of 375 samples as opposed to 48,000 for motion generated from audio sources. This can be a great savings in terms of memory and CPU usage.

Motion FX objects can contain one type of plug-in sources:

- **Signal generator source** - That generates motion from a special signal generator, such as the tone generator or Wwise motion generator.

The source you choose will depend on the device you are creating the motion for and the type of motion you want. Each source has a series of properties that you can use to tweak the motion effect.



Note

Before creating motion FX objects, you must enable "Game Controller" on the Motion Devices tab of the Project Settings dialog box.

Creating Motion FX Objects

Motion FX objects represent the motion assets in your project. They are the foundation for the motion structures within the **Actor-Mixer Hierarchy** and can be grouped into containers and actor-mixers. Like sound objects, they contain sources that are either linked to media files or use signal generator plug-ins. A series of properties and behaviors exist for motion objects allowing you to create realistic and timely feedback for the game player.

You can create any number of motion FX objects in the **Actor-Mixer Hierarchy** and then use containers and actor-mixers to build the motion structure that your project requires. Motion FX objects can be created in two ways:

- **Manually** - By creating a motion FX object in the Actor-Mixer hierarchy.
- **Automatically** - By importing a media file as a source of a motion FX object.

To create motion FX objects:

1. In the Audio tab of the Project Explorer, right-click the work unit in which you want to create a motion FX object.
2. From the shortcut menu, select **New Child > Motion FX**.

The new motion FX object is added to the Actor-Mixer hierarchy.



Note

The first time you create a motion FX object, a message box is displayed prompting you to enable a motion devices for your

project. The motion FX object will be created regardless, but you won't be able to create any sources until a motion device is enabled. For more information on enabling motion devices, refer to [Enabling the Motion Devices for Your Project](#).

3. Replace the default name with one that best represents the object.



Note

The following characters may not be used when naming motion FX objects in Wwise: ':<>%*?/\|.'

Related Topics

- [Creating Sources for Motion FX Objects](#)
- [Generating Motion for Game Controllers](#)
- [Generating Motion from Existing Sounds](#)

Creating Sources for Motion FX Objects

To give you additional control and flexibility over the type of motion that is generated by your game, Wwise offers a number of different plug-ins that can be used to generate motion.

Creating plug-in sources for motion FX objects is the same as it is for sound objects.

To create plug-in sources for motion FX objects:

1. Load a motion FX object into the Property Editor.
2. In the Contents Editor, click **Add Source** in the motion device title bar.

The Source menu is displayed with a list of source plug-ins that are available for the current motion device.

3. Click the source plug-in that you want to add.

The plug-in source is added to the motion FX object and appears as a new entry for the selected motion device.

4. Double-click the source plug-in to display a complete list of properties in the Property Editor.
5. Edit the properties as necessary.



Note

For a complete description of the source plug-ins properties, click the Help button in the property editor.

Related Topics

- [Creating Motion FX Objects](#)
- [Generating Motion from Existing Sounds](#)

Generating Motion for Game Controllers

Since game controllers are only capable of re-creating a limited range of motion effects based on the speed of the motors, a special source plug-in, called the Wwise Motion Generator, was designed to generate motion for these types of devices. This special plug-in allows you to create and tailor one or more motion curves for each platform's controller. Each motion curve defines the intensity of the motion effect over a specific period of time with the option of using ADSR envelope controls to further define the shape of the curve.

By default, each motion curve has an intensity value of 1 (or 100%) for the entire time period, which results in full motion being generated. You can move the existing control points and add new ones to create the desired effect.

To create a more detailed and complex motion curve, you can also define the shape of each curve segment. A curve segment is any part of the curve between two control points. You can choose from a variety of curve shapes, including linear, constant, logarithmic, exponential, and s-curve. For more information on specifying curve shapes and other information about working in the graph view, refer to [Chapter 42, Getting to Know the Graph View](#).

You can create different curves for each of the following:

- **Default Small** - A default curve that can be used by the small motor of one or more controllers.
- **Default Large** - A default curve that can be used by the large motor of one or more controllers.
- **Xbox 360 Small** - A curve that is assigned to the small motor of the Xbox 360 controller.
- **Xbox 360 Large** - A curve that is assigned to the large motor of the Xbox 360 controller.
- **PlayStation 3 Small** - A curve that is assigned to the small motor of the PlayStation 3 controller.

- **PlayStation 3 Large** - A curve that is assigned to the large motor of the PlayStation 3 controller. The PS3 Move uses this curve only.
- **Wii** - A curve that is assigned to the motor in the Wii remote.
- **Xbox One Small** - A curve that is assigned to the small motor of the Xbox One controller.
- **Xbox One Large** - A curve that is assigned to the large motor of the Xbox One controller.
- **PlayStation 4 Small** - A curve that is assigned to the small motor of the PlayStation 4 controller.
- **PlayStation 4 Large** - A curve that is assigned to the large motor of the PlayStation 4 controller.

You may, however, want to re-use some of the curves to save on time. You can also solo one motor at a time to more easily fine-tune its motion curve. When a motor is soloed, all other motors are muted during playback.

To create motion effects using the Wwise Motion Generator plugin:

1. Load a motion FX object into the Property Editor.
2. In the Contents Editor, click the **Add Source** button in the title bar of the Controller motion device.
3. From the list, click **Wwise Motion Generator**.

A motion source is added under the Controller title bar.

4. Double-click the source to load it into the Property Editor.
5. From the Curve list, select the **Default Small** curve.

The curve is displayed in the graph view.

6. Manipulate the Default Small motion curve by doing any of the following:

Add points along the curve.

Drag points to a new location or type specific values into the X and Y coordinate boxes.

Define the shape of each curve segment.



Note

To fine-tune the motion of a particular motor, you can solo each motor in the list. When a motor is soloed, all other motors are muted during playback.

7. For the remaining curves, do one of the following:

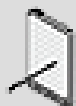
Select **Custom** from the Usage list and create a new motion curve.

Re-use an existing curve by selecting the curve name from the Usage list.

8. In the Period text box, specify the amount of time to complete one motion curve cycle.

This value defines the length of the X-axis.

9. If you want to modify the period of the motion effect in real time using an RTPC, you must type a value in the Period Multiplier text box. When you change the period multiplier value, the curve will either be stretched or shrunk to fit the new time period.



Note

If no RTPC is used, the period multiplier should be kept at its default value of 1.

10. To specify the duration of the motion effect, select one of the following:

One period - To use one period as the duration of the motion effect, where a period is defined as the period multiplied by the period multiplier.

Fixed duration - To specify a fixed duration for the motion effect. Depending on the value specified for the “Fixed duration”, the motion effect may complete any number of curve cycles. If the fixed duration value is not a multiple of the period multiplied by the period multiplier, the last curve cycle will be incomplete.

Envelope - To use ADSR envelope controls to determine the shape of the curve over time.

Related Topics

- [Generating Motion from Existing Sounds](#)

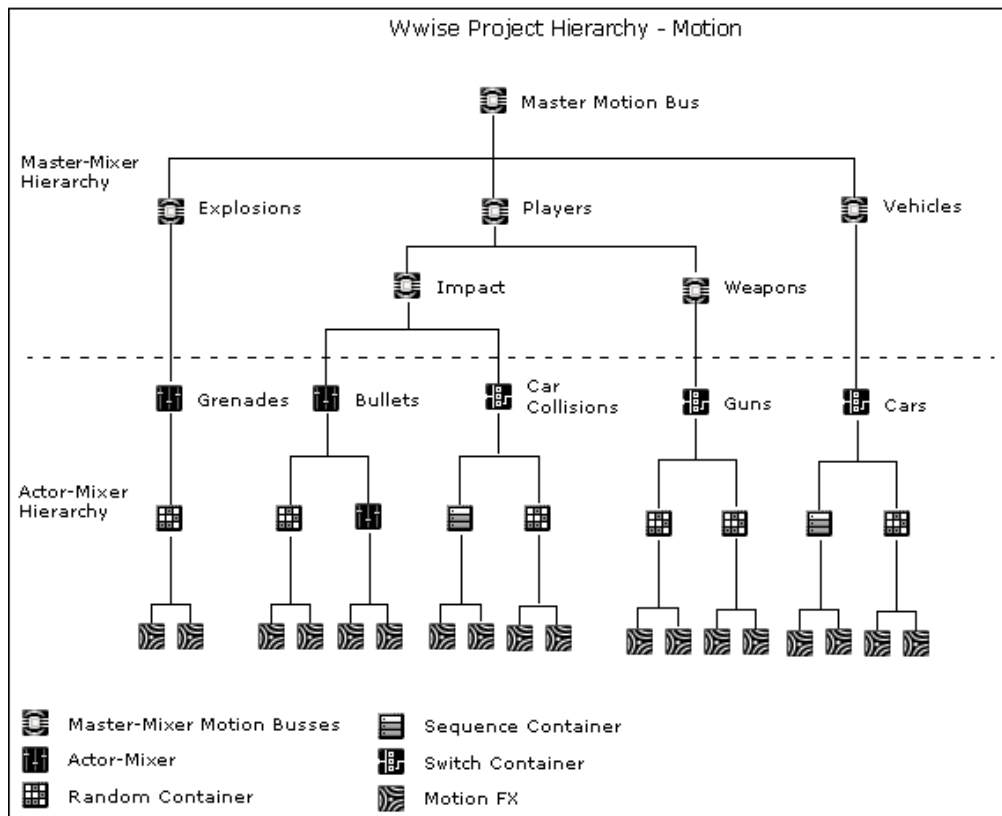
Building an Output Structure for Motion

Since motion data has a separate processing pipeline than audio, it requires its own structure of busses. To handle both motion and audio, the Master Mixer hierarchy has been divided into two main branches: one for audio and one for motion. At the top of each of these branches are the following master busses:

- **Master Audio Bus** - A top-level output bus where the final mix for audio data is performed.

- **Master Motion Bus** - A top level output bus where the final mix for motion data is performed.

You can create an unlimited number of busses under each master bus to re-group the many different sounds, music, and motion objects in your project. When building a bus structure, the same features, rules, and conditions apply to both branches. Since audio and motion are handled differently, all sounds and music objects must pass through the Master Audio Bus and all motion data must pass through the Master Motion Bus.



For more information on building an output structure for motion, refer to [Overview](#).

Motion Tips and Best Practices

Before creating the motion for your game, you may want to review the following sections, which provide you with a series of examples, tips, and best practices that can help you better manage the motion in Wwise.

Performance

If you have the time, you should create and integrate motion into your game using motion FX objects instead of generating it directly from your existing

sounds because it can save on processing power because the motion data uses fewer samples and doesn't need to be filtered and re-sampled at run-time.

Motion Devices

- Game controllers have limited playback capability. Game controllers can play back only a limited range of motion effects, making it difficult for a game player to distinguish between different types of motion effects especially when several effects are played back simultaneously. It is a good idea, therefore, to try to limit the playback to one motion source at a time.
- Connect game controllers to high power USB ports. If your motion objects are not playing back on a game controller, make sure that the game controller is connected to a high power USB port. If the USB port does not have sufficient power to run the motion device, the system unmounts the device to protect the operating system or the device itself. The USB ports in the front of a computer are generally not powerful enough to run a motion device, so you should connect them to the USB ports at the back of the computer.

Game Controllers

There may be situations where you want the duration of the motion generated by the small and large motors of the Xbox 360 or PlayStation 3 controllers to be different. Since the same duration controls in the Wwise Motion Generator plug-in are applied to all curves, this type of motion effect can't be created using one source. To work around this limitation, you can do the following:

- Create 2 motion FX objects, each with a Wwise Motion Generator as a source.
- For one motion FX object, generate the curve for the large motor, define the duration for this curve, and then make the small motor curve flat at 0.
- For the other motion FX object, generate the curve for the small motor, define the duration for this curve, and then make the large motor curve flat at 0.
- Trigger both motion FX objects using the same event.

When playing back the event in Wwise, you can also use the PF exclusion option in the Event Editor as a solo/mute button.

Troubleshooting Motion

If the motion created in Wwise is not playing on a particular motion device, you can troubleshoot the problem using the Profiler. To capture motion data in the Advanced Profiler, do the following:

- Press F6 to switch to the Profiler layout.
- From the menu bar, click **Project > Profiler Settings**.
- Select the **Motion Data** option and click OK.

The Motion Devices tab is added to the Advanced Profiler.

Now you can connect to your game and profile the motion to identify the potential problem. The “Motion Devices” tab contains two separate lists: one for Devices and one for Game Objects. The Devices list indicates which motion devices have been properly registered and initialized in the sound engine, for each player.

To troubleshoot the problem, first check to see if the motion devices have been properly initialized, by doing the following:

- Verify that each motion device is displayed in the Device list. If the device doesn't show up in this list, then the call to `AK::MotionEngine::RegisterMotionDevice` failed.

If the motion device appears in the list, then check to see if the motion device has been associated with a particular player, by doing the following:

- Verify that there is a check mark matching up the device to the appropriate player. If there is no check mark for a particular player, then either of the following may be true:
 - The call to `AK::MotionEngine::AddPlayerMotionDevice` failed.
 - `AK::MotionEngine::RemovePlayerMotionDevice` was called.

If the motion devices have been properly initialized and they are associated to the appropriate player, then you can check the Game Objects list to verify if the game objects are setup properly to send motion data to a specific player. If there is no check mark for a particular combination of game object/player, you can switch to the Listener tab of the Advanced Profiler to check the following:

- In the Listener list, find the player number that is not receiving the motion data in the Players column and then check to see which listener the player is using to receive the motion data. If the player is not using the correct listener, verify your calls to `AK::MotionEngine::SetPlayerListener`.
- If the player is using the right listener, then make sure that the listener to which the player is attached has a check mark in the Motion column. If not, verify that you are calling `AK::SoundEngine::SetListenerPipeline` properly.
- If a check mark is displayed, check the Game Object list to verify that the listener used by the player is enabled for the game object on which the Motion FX is playing. If the listener is not enabled for the game object, verify your calls to `AK::SoundEngine::SetActiveListeners`.

audio**kinetic**

Part IV. Interacting with the Game



14. Managing Events	364
Overview	365
Creating Events	370
Working with Events	378
Event Tips and Best Practices	382
15. Managing Dynamic Dialogue	384
Overview	385
Understanding the Dynamic Dialogue System	385
Creating Dialogue Events	389
Working with Dialogue Events	400
Dialogue Events Tips and Best Practices	404
16. Working with States	407
Overview	408
Working with States	409
Defining Transitions Between States in a State Group	413
Assigning States to Objects and Busses	414
States Tips and Best Practices	420
17. Working with Switches	421
Overview	422
Working with Switches	423
Mapping Game Parameter Values to Switches	426
Switches Tips and Best Practices	427
18. Working with RTPCs	429
Overview	430
Managing Game Parameters Used in RTPCs	431
Controlling Property Values Using Game Parameters	436
Working with LFOs	443
Working with Envelopes	445
Viewing Game Objects	447
RTPC Tips & Best Practices	447
19. Working with Triggers	449
Overview	450
Working with Triggers	451
20. Working with States and State Groups for Dynamic Dialogue	453
Overview	454
Working with State Groups	455

Chapter 14. Managing Events

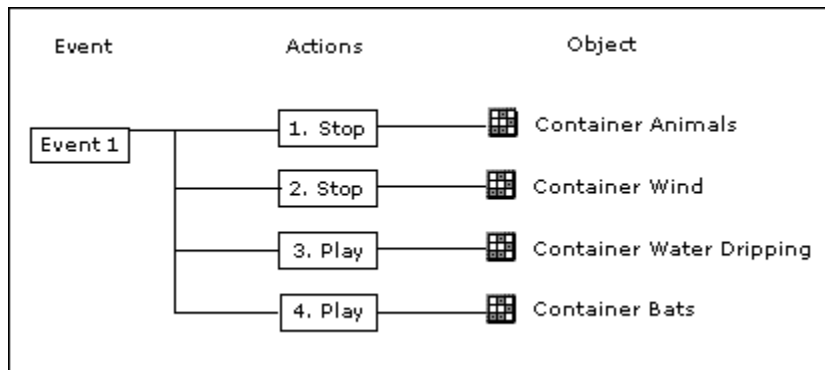
Overview	365
Creating Events	370
Working with Events	378
Event Tips and Best Practices	382

Overview

Wwise uses events to drive the sound, music, dialogue, and motion in-game. Events determine which sound, music, motion, or piece of dialogue is played at any particular point in the game. To accommodate as many situations as possible, there are two kinds of events:

- “Action” events
- Dialogue events

“Action” events contain one or more actions that are applied to the different sound, music, or motion structures within your project hierarchy. The actions you select will specify whether the Wwise objects will play, pause, stop, and so on. For example, the following event could describe a point in a game where a character leaves a windswept field to enter an eerie cave.



Note


In most cases, a Play event is paired with a Stop action event or even a Pause or Resume event. As an alternative to creating separate events for these types of actions, you can use the Wwise SDK to execute these actions on the sound. For more information, refer to [Stopping, Pausing, and Resuming Sounds Programmatically](#).

Dialogue events, on the other hand, use a type of decision tree containing state groups to dynamically determine what object is played. For more information about dialogue events, refer to [Chapter 15, Managing Dynamic Dialogue](#).

After you create “action” events in Wwise, they can be integrated into the game engine so that they are called at the appropriate times in the game. Since the game engine uses the event name or ID, you can create the events, integrate them into the game and then build and fine-tune the contents of the event

without ever having to re-integrate it into the game again. As long as the event name or ID does not change, no additional programming is required. This gives you a great deal of flexibility to experiment with different sounds or motion objects, add or remove objects, and fine-tune transitions between objects.

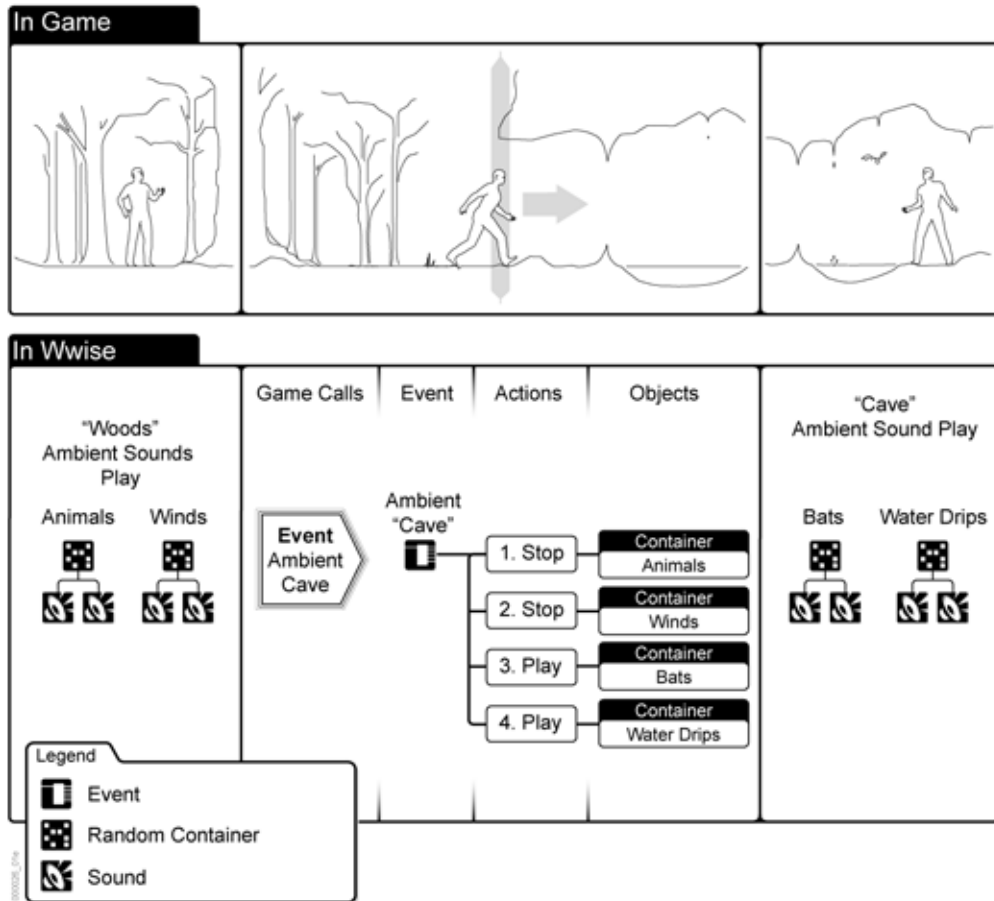
To help you easily identify an event in the interface, it is represented by the following icon.

Icon	Represents
	Event

Using Events - Example

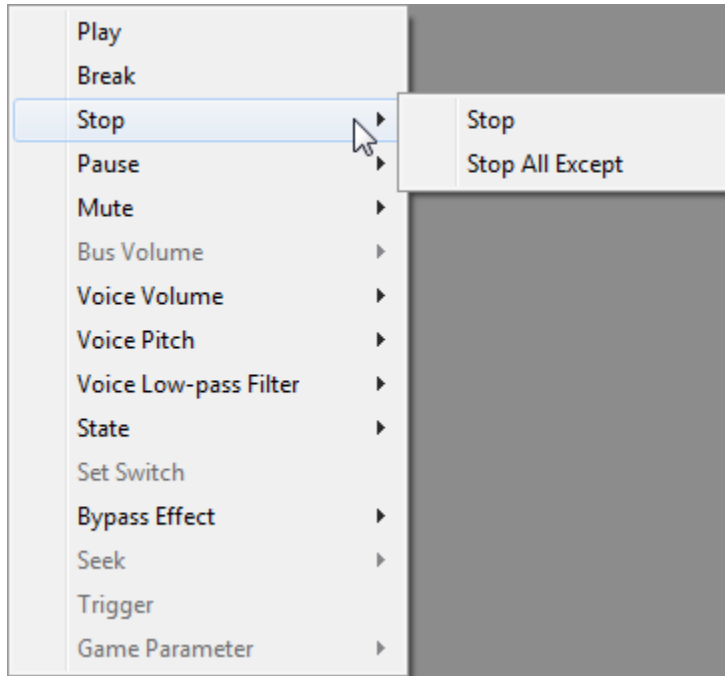
Let's say you are creating a fantasy role-playing game. You know that in one level of the game, the character will enter a cave from the woods. You want the ambient sounds to change at the moment the character enters the cave. At the beginning of your project, you can create an event in Wwise using temporary or placeholder sounds. The event will contain a series of actions that will stop the ambient "Woods" sounds and play the ambient "Cave" sounds. After the event is created, it can be integrated into the game so that it will be triggered at the appropriate moment. Since no additional programming is required after the initial integration, you can experiment with different sounds, add and remove actions, and change action properties until it sounds just right.

The following illustration shows the relationship between the action in-game, the event, and the sounds that are played.

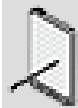


Types of Event Actions

Wwise comes with a variety of actions that you can use to drive the sound, music, and motion in game. The actions are grouped by category and each category contains a series of actions that you can choose from.



Each action also has a set of properties, such as delays and fades, that you can use to better manage the incoming and outgoing objects. The following table describes each of the event actions.



Note

The following descriptions are general descriptions and do not take into account the scope setting for each action. The scope, which is either Game Object or Global, specifies the extent to which the action is applied to objects within the game.

Event Action	Description
Play	Plays back the associated object.
Post Event	Triggers an event from within another event.
Break	Stops playback of a looped sound or continuous container while allowing the current object or objects to finish playing.
Stop	Stops playback of the associated object.
Stop All	Stops playback of all objects.
Stop All Except	Stops playback of all objects except those specified.
Pause	Pauses playback of the associated object.
Pause All	Pauses playback of all objects.
Pause All Except	Pauses playback of all objects except those specified.
Resume	Resumes playback of the associated object that had previously been paused.
Resume All	Resumes playback of all paused objects.

Event Action	Description
Resume All Except	Resumes playback of all paused objects, except those specified.
Mute	Silences the associated object.
Unmute	Returns the associated object to its original “pre-silenced” volume level.
Unmute All	Returns all objects to their original “pre-silenced” volume levels.
Unmute All Except	Returns all objects, except those specified, to their original “pre-silenced” volume levels.
Set Volume	Changes the volume level of the associated object.
Reset Volume	Returns the volume of the associated object to its original level.
Reset Volume All	Returns the volume of all objects to their original levels.
Reset Volume All Except	Returns the volume of all objects, except those specified, to their original levels.
Set Pitch	Changes the pitch for the associated object.
Reset Pitch	Returns the pitch of the associated object to its original value.
Reset Pitch All	Returns the pitch of all objects to their original values.
Reset Pitch All Except	Returns the pitch of all objects, except those specified, to their original values.
Set Low-pass Filter	Changes the amount of low pass filter applied to the associated Wwise object. Low-pass Filter does not affect motion objects.
Reset Low-pass Filter	Returns the amount of low pass filter applied to the associated Wwise object to its original value. Low-pass Filter does not affect motion objects.
Reset Low-pass Filter All	Returns the amount of low pass filter applied to all Wwise objects to their original values. Low-pass Filter does not affect motion objects.
Reset Low-pass Filter All Except	Returns the amount of low pass filter applied to all Wwise objects, except the ones that you specified, to their original values. Low-pass Filter does not affect motion objects.
Set High-pass Filter	Changes the amount of high-pass filter applied to the associated Wwise object. High-pass Filter does not affect motion objects.
Reset High-pass Filter	Returns the amount of high-pass filter applied to the associated Wwise object to its original value. High-pass Filter does not affect motion objects.
Reset High-pass Filter All	Returns the amount of high-pass filter applied to all Wwise objects to their original values. High-pass Filter does not affect motion objects.

Event Action	Description
Reset High-pass Filter All Except	Returns the amount of high-pass filter applied to all Wwise objects, except the ones that you specified, to their original values. High-pass Filter does not affect motion objects.
Set State	Activates a specific State.
State > Enable State	Re-enables the State or States applicable to the associated Wwise object after having applied a Disable State Action.
State > Disable State	Disables the State or States for the associated Wwise object.
Set Switch	Activates a specific Switch.
Enable Bypass	Bypasses the effect(s) applied to the associated Wwise object.
Disable Bypass	Removes the effect bypass which re-applies the effect(s) to the associated Wwise object.
Reset Bypass Effect	Returns the bypass effect option of the associated object to its original setting.
Reset Bypass Effect All	Returns the bypass effect option of all Wwise objects to their original settings.
Reset Bypass Effect All Except	Returns the bypass effect option of all Wwise objects, except the ones that you specified, to their original settings.
Trigger	Calls a trigger that, in turn, launches a stinger.
Seek	Changes the playback position of the associated Wwise object. It has no effect on objects that are not currently playing.
SeekAll	Changes the playback position of all Wwise objects. It has no effect on objects that are not currently playing.
Release Envelope	Releases envelopes associated to the Wwise object.
Reset Playlist	Reset playlist of the specified random/sequence container to the initial state. This has no impact on the continuous mode playback nor on currently playing sounds.

Creating Events

Every sound, motion object, or piece of music in your game is driven by an event. The event creation process involves the following steps:

- [Creating a New Event](#)
- [Adding Actions to an Event](#)
- [Assigning Objects to Event Actions](#)
- [Defining the Scope of an Event Action](#)
- [Setting Properties for an Event Action](#)

To give you additional control and flexibility, events can either perform one action or a series of actions. The management of events is done using the Event Editor.

When authoring across platforms, you may want to exclude certain actions from a specific platform. By default all actions are included in an event, but you can customize this per platform. For more information on authoring across platforms, refer to [Excluding Project Elements from a Platform](#).

If you are working as part of a team on the same project, you can assign the events to different work units so that each member of the team can work on different events simultaneously. For more information on working with work units, refer to [Dividing Your Project into Work Units](#).

Creating a New Event

When creating a new event, you can do any one of the following:

- Create an empty event that includes no actions or objects.
- Create an event that includes a particular action.
- Create an event that includes both an action and an object.



Tip

You can also create events by right-clicking in the Event viewer. When adding events from the Event viewer, you must also assign the event to a particular work unit.

To create an empty event:

1. In the Project Explorer, switch to the **Events** tab.
2. Do one of the following:

Select a work unit or virtual folder and click the **Event** icon in the Project Explorer toolbar.

Right-click the work unit or virtual folder and select **New Child > Empty Event** from the shortcut menu.

A new event is created within the work unit or virtual folder you selected in the Project Explorer.

3. Replace the default name with one that best represents the event.



Note

Event names can contain only letters, numbers, and underscores. They must also start with either a letter or underscore.

To create an event with an action:

1. In the Project Explorer, switch to the **Events** tab.
2. Right-click the work unit or virtual folder to which you want to add the event.
3. From the shortcut menu, select **New Child** to display the event action list.
4. From the action list, select an action category.

A submenu with a series of actions is displayed.

5. Select an action from the list.

A new event that includes the action you selected is created within the work unit you selected in the Project Explorer.

6. Replace the default name with one that best represents the event.



Note

Event names can contain only letters, numbers, and underscores. They must also start with either a letter or underscore.

To create an event or events that include both an action and object:

1. From the Audio tab in the Project Explorer, select one or more objects that you want to include in the event and then right-click the selection.

A shortcut menu is displayed.

2. Select one of the following options:

New Event to create an event that includes the selected object.

New Event (Single event for all objects) to create one event that includes all the selected objects.

New Events (One event per object) to create one event for each object selected.

3. From the action list, select an action category.

A submenu with a series of actions is displayed.

4. Select an action from the list.

One or more events are created in the Event Editor with the selected action and object(s).

5. In the Name field, replace the default name with one that best represents the event.



Note

Event names can contain only letters, numbers, and underscores. They must also start with either a letter or underscore.

Related Topics

- [Adding Actions to an Event](#)
- [Assigning Objects to Event Actions](#)
- [Defining the Scope of an Event Action](#)
- [Setting Properties for an Event Action](#)

Adding Actions to an Event

You must define the actions that you want to be included in your events. Each event can contain one or several actions.

To add an action to an event:

1. In the Event Editor, click the Action Selector button (>>).

A list of actions is displayed.

2. From the action list, select an action category.

A submenu with a series of actions is displayed.

3. Select an action from the list.

The action you selected is added to the event.

You can now assign an object to the event action or continue to add actions to the event.

Related Topics

- [Types of Event Actions](#)
- [Creating a New Event](#)
- [Assigning Objects to Event Actions](#)
- [Defining the Scope of an Event Action](#)
- [Setting Properties for an Event Action](#)
- [Playing Back an Event](#)

Assigning Objects to Event Actions

Most event actions must be assigned to a particular object, structure, or game sync. Events that contain one or more actions without an associated object are called orphaned events. These orphaned events will appear in the Orphans tab of the Event Viewer or when you generate an integrity report for your project.

To help you identify the status of objects within an event, the object name will appear in one of the following colors:

- **White** - For objects included in the current platform.
- **Gray** - For objects not included in the current platform.
- **Red** - For event actions that are missing an associated object or objects that are missing from the current project.
- **Yellow** - For objects that are currently unloaded from the current project.



Note

When assigning a music object to certain actions, such as Trigger, Set Switch, and Set State, be aware that the result of these actions may be delayed due to pre-defined points in the music object where these specific actions can occur.

To assign an object to an event action:

1. In the Event Editor, select the action to which you want to assign an object.
2. Click **Browse**.

The Project Explorer - Browser is displayed.

3. Navigate through the hierarchy and select the object that you want to assign to the action.
4. Click **OK**.

The object is assigned to the action.



Tip

You can also assign an object to an event action by dragging it from the Project Explorer to the action in the Event Actions list.

Related Topics

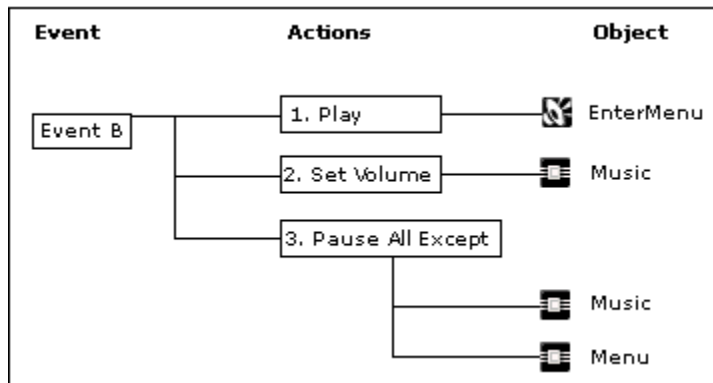
- [Creating a New Event](#)

- [Adding Actions to an Event](#)
- [Defining the Scope of an Event Action](#)
- [Setting Properties for an Event Action](#)
- [Playing Back an Event](#)

Defining the Scope of an Event Action

When creating an event, you must define the scope for each action. The scope specifies the extent to which the action is applied to objects within the game. The scope can either be applied globally to all game objects or the specific game object that triggered the event. For some actions, you can choose the scope, and for other actions, the scope is pre-determined.

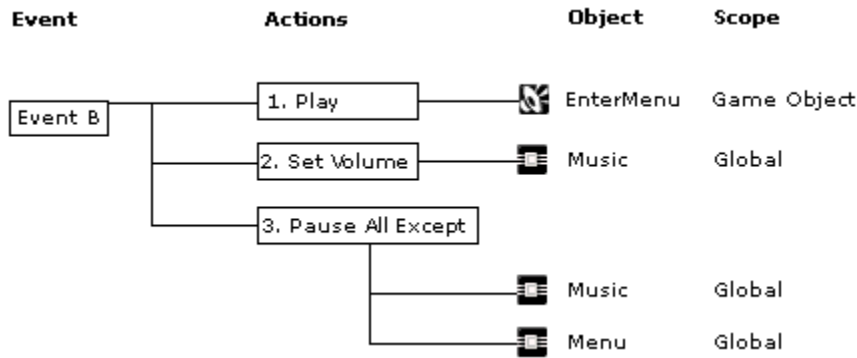
For example, let's say you created an event for when the player leaves the game to enter the menu. This event will play the “Enter_Menu” sound, decrease the volume of the music bus by -10 db, and pause everything else. The following illustration demonstrates how this event would look in Wwise.



The scope for each of these event actions would be as follows:

Event Action	Scope	Comments
Play > Menu_Enter	Game Object	The scope is set to Game Object because play events are always triggered by a single game object.
Set Volume > Music	Global	The scope is set to Global because the Set Volume action is applied to a bus, which, by its very nature, is global.
Pause All Except > Music	Global	The scope is automatically set to Global because the Pause All Except action is applied to the music bus, which, by its very nature, is global.
Pause All Except > Menu	Global	The scope is automatically set to Global because the Pause All Except action is applied to the menu bus, which, by its very nature, is global.

As you can see in the previous example, if you use one of the “Except” actions, you must define the scope for each object in the exceptions list.



To define the scope of an event:

1. From the Scope list, select one of the following options:

Game object to apply the event action to the game object that triggered the event.

Global to apply the event action to all game objects.

Related Topics

- [Creating a New Event](#)
- [Adding Actions to an Event](#)
- [Assigning Objects to Event Actions](#)
- [Setting Properties for an Event Action](#)
- [Playing Back an Event](#)

Setting Properties for an Event Action

Each event action has a set of related properties that you can use to further refine your game's sound, music, or motion.

Each action contains different properties, but they all fall into one of the following categories:

- Delays
- Transitions
- Bypass effect properties
- Property settings for volume, pitch, LPF, states, or switches.



Note

When setting the delay properties of certain event actions, such as Trigger, Set State, and Set Switch, that involve a music object, be aware that the actual delay may be longer than the one

specified due to pre-defined points in the music object where these specific actions can occur.

To set the action properties of an event:

1. In the Event Editor, select an action from the Event Actions list.

The properties associated with the selected action are displayed in the Action Properties group box.

2. Specify the values for the following properties, where appropriate:

Delay

Transition

Effect bypass settings

Volume, pitch, LPF, state, or switch settings.

Related Topics

- [Creating a New Event](#)
- [Adding Actions to an Event](#)
- [Assigning Objects to Event Actions](#)
- [Defining the Scope of an Event Action](#)
- [Playing Back an Event](#)

Playing Back an Event

At any point in the creation process, you can audition an event.



Note

To audition motion, the corresponding motion device must be connected to your workstation.

To play back an event:

1. Do one of the following:

Select an event in the Event Viewer.

Load an event into the Event Editor.

The event is loaded into the Transport Control.

2. Click the Play icon in the Transport Control.

The event is played back.



Note

You can also play back events using the Soundcaster. For more information on using the Soundcaster, refer to [Auditioning Sounds, Music, and Motion FX in the Soundcaster](#).

Related Topics

- [Creating a New Event](#)
- [Adding Actions to an Event](#)
- [Assigning Objects to Event Actions](#)
- [Defining the Scope of an Event Action](#)
- [Setting Properties for an Event Action](#)

Working with Events

Because no additional programming is required after the initial integration of an event, you can experiment with different sounds or motion objects, change the properties of existing objects, add and remove actions, and change action properties until everything is exactly as you want it.

When managing events, you can perform the following tasks:

- [Renaming an Event](#)
- [Removing Actions from an Event](#)
- [Replacing Objects Assigned to Event Actions](#)
- [Deleting Events](#)

Renaming an Event

Wwise automatically gives a name to an event when it is created. It is good practice to rename the event with a name that is more descriptive. Each event name must be unique, and consist only of letters, digits, and underscores. The first character must also be either a letter or an underscore.

Unless absolutely necessary, you shouldn't rename an event after it has been integrated into the game because it will require additional programming.

To rename an event:

1. In the Event Editor, click in the Name field.

The name of the event is highlighted.

2. Type a new name for the event.



Note

You can also rename events on the Events tab of the Project Explorer.

Related Topics

- [Creating a New Event](#)
- [Removing Actions from an Event](#)
- [Replacing Objects Assigned to Event Actions](#)
- [Deleting Events](#)

Removing Actions from an Event

While you are experimenting and building your events, you may need to remove one or more actions from an event. As long as the event name doesn't change, you can remove actions without any additional programming required.

To remove actions from an event:

1. In the Event Editor, right-click the action that you want to remove from the event.

A shortcut menu is displayed.

2. From the menu, select **Remove Action**.

The action is removed from the event.



Tip

You can also remove actions from an event by selecting the action and pressing the Delete key.

Related Topics

- [Adding Actions to an Event](#)
- [Renaming an Event](#)
- [Replacing Objects Assigned to Event Actions](#)
- [Deleting Events](#)

Replacing Objects Assigned to Event Actions

You may want to replace specific sounds, music, or motion objects with different ones to see how they will fit in game. Even after an event is integrated into the game, you are free to experiment with different sounds, music, motion, actions, and so on.

To replace an object that is assigned to an action:

1. In the Event Editor, select the action whose object you want to replace.
2. Click **Browse**.

The Project Explorer - Browser is displayed.

3. Navigate through the hierarchy and select the new object that you want to assign to the action.
4. Click **OK**.

A confirmation message is displayed.

5. Click **Replace**.

The new object is assigned to the action.



Tip

You can also replace an object that is assigned to an action by dragging a new object from the Project Explorer to the event action in the Event Editor. A confirmation message is displayed before the object is replaced.

Related Topics

- [Assigning Objects to Event Actions](#)
- [Creating a New Event](#)
- [Deleting Events](#)

Displaying an Event's Object in the Schematic View

If you want to view the project structure or pipeline for a particular object that is included in an event, you can quickly display it in the Schematic view.

To display the event object's pipeline in the Schematic view:

1. In the Event Editor, right-click the object whose pipeline you want to view.

A shortcut menu is displayed.

2. Click **Show in Schematic View**.

The sound, music, or motion pipeline is displayed in the Schematic view.

Related Topics

- [Replacing Objects Assigned to Event Actions](#)
- [Assigning Objects to Event Actions](#)
- [Creating a New Event](#)
- [Deleting Events](#)
- [Playing Back an Event](#)

Deleting Events

If you no longer need an event, you can delete it. Before deleting an event, you may want to verify if it is being used by another member of your team in different part of the project or if it is already included in one of your SoundBanks. If you or someone else on your team deletes an event that is included in a SoundBank, it will create an invalid event. Wwise doesn't automatically remove deleted events or other invalid project elements from SoundBanks, so you will have to remove them manually. To help you locate these types of events and object structures within a SoundBank, Wwise adds the word “Missing” after their name on the Add tab of the SoundBank Editor. For more information on invalid events, refer to [Removing Project Elements from a SoundBank](#).

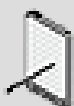
To delete an event:

1. Do one of the following:

On the Events tab of the Project Explorer, right-click the event you want to delete and select **Delete Selection** from the shortcut menu.

In the Event Viewer, click the event you want to delete and press the **Delete** key.

The event is deleted.



Note

If you delete an event by mistake, you can undo the delete, by pressing **Ctrl+Z** or by clicking **Edit > Undo**.

Related Topics

- [Creating a New Event](#)

- [Removing Actions from an Event](#)
- [Renaming an Event](#)
- [Searching for Elements within a SoundBank](#)

Event Tips and Best Practices

Before creating events, you may want to review the following sections, which provide you with a series of tips and best practices that can help you better manage the events in your project.

Re-naming Events

Before changing the name of an event, verify how the events have been integrated into the game. If events have been integrated using strings containing the name of the event or IDs from the `Wwise_IDs.h` header file, changing the name will require additional programming to be able to use the event.

Organizing Events

You can use work units and virtual folders to organize the events in your project. Dividing your events into multiple work units can be very useful when several people are working in the project at the same time. For example, each team-member can create their events in their own personal event work unit. Dividing up your events into different work units should not affect your productivity or workflow, but by doing so, you can avoid merge conflicts that can be both frustrating and time-consuming to fix.

Matching Game Events to Events in Wwise

You can use the SoundBank definition file to track which events have already been integrated into your game, which ones are missing, and which ones still need to be created in Wwise. The audio programmer can generate the list of events from the game and then you can import the definition file into Wwise. You can use the information in the log file to match up the events in game with those created or missing from Wwise. For more information on SoundBank definition files, refer to [Populating a SoundBank by Importing a Definition File](#).

Stopping, Pausing, and Resuming Sounds Programmatically

Event information is stored within the Default Memory pool of the sound engine. To avoid using too much space within the default memory pool, you can stop, pause, and resume sounds programmatically using the `ExecuteActionOnEvent` function in the Wwise SDK. Instead of creating Play/Stop event pairs, you can create a single Play event with a call to `ExecuteActionOnEvent` to stop the sound. If you need to pause and resume the sound, you don't need to create separate events, you can simply use the

same function to execute the pause and resume actions. This can significantly reduce the number of events within your project and can free up some valuable space in the Default Memory pool. To give you additional control, you can also specify fade out times when stopping, pausing, or resuming sounds programmatically. For more information about the `ExecuteActionOnEvent` function, refer to [the Wwise SDK documentation](#).

Using SetPitch Events to Create Pitch Envelopes

For most natural sounds, pitch actually fluctuates or changes for a time until it reaches its steady state or 'sustain' level. To create this effect, you can use a pitch envelope. Now Wwise does not support pre-programmed pitch envelopes, but you can get close to that behavior by creating an event with multiple consecutive `setPitch` actions. Each set-pitch action in the event could adjust the pitch over a period of time following one of the predefined curves in the Action properties. Of course, the last action in the sequence would probably need to be a `ResetPitch` in order to get the game object to its original state.

Chapter 15. Managing Dynamic Dialogue

Overview	385
Understanding the Dynamic Dialogue System	385
Creating Dialogue Events	389
Working with Dialogue Events	400
Dialogue Events Tips and Best Practices	404

Overview

Many games today, including real-time strategy, sports, and adventure games, have an audio component that is dynamic or driven by the action that is taking place in game. With all the different variables and outcomes that can exist in a game, this type of audio certainly represents a unique challenge for scriptwriters and audio designers. To cover all the different combinations of dialogue for each condition or outcome, you may need thousands of assets and complex switch container hierarchies. This can be very costly in terms of memory usage. In an effort to streamline the development process and reduce the overall memory consumption, Wwise introduces the Dynamic Dialogue system.

The Dynamic Dialogue system is a lightweight and efficient method for building and managing dynamic audio in Wwise. It uses a set of rules within a decision-tree structure to determine which piece of dialogue to play at any particular moment in game.

Additional features, available in the Wwise SDK, allow you to build on this already powerful system by creating runtime dynamic sequences, which are ideal for play-by-play commentary in sports games. In Wwise, you can break down dialogue into individual words or small phrases and then assign these pieces of dialogue to specific paths within the decision-tree structure. As the game is being played, the specific words are submitted to the sound engine where they are dynamically stitched together to form free-flowing sample accurate sentences. Because words and small phrases exist separately, they can also be combined and re-used in a variety of ways. For more information on integrating dynamic sequences into your game, refer to the [Wwise SDK documentation](#).



Note






Although dialogue events were initially created to handle game dialogue, they are not reserved explicitly for dialogue and can be used for a variety of other purposes in your game, including footstep sounds on a variety of ground surfaces.

Understanding the Dynamic Dialogue System

At the center of Wwise's Dynamic Dialogue system is the dialogue event, which is basically a set of rules or conditions that determines which piece of dialogue to play. The dialogue event allows you to re-create a variety of different scenarios, conditions or outcomes that exist in your game. To ensure that you cover every situation, Wwise also allows you to create default or fallback conditions.

All these conditions are defined using a series of state groups and states. These state groups and states are combined to create paths, which define the particular conditions or outcomes in the game. Each path is then associated with a specific sound object in Wwise. As the game is played and dialogue events are called, the game verifies the existing conditions against the paths defined in the dialogue event. The paths that match the current situation in game along with the mode, probability, and weighting of each path determine which piece of dialogue is played, if any.

For example, the following dialogue event contains the state groups related to the names of each player in a sports game. The values of each state group are combined to create the different paths or conditions that may exist. In this particular example, the commentator can use either the player's last name or full name.

Dialogue Event: Name		
Arguments	Player Name	Name Length
Argument Values	Tony Cross	Full
	John Patrick	Last
Argument Paths	Assigned Object	
Cross - Full	 Cross_Full	
Cross - Last	 Cross_Last	
Patrick - Full	 Pat_Full	
Patrick - Last	 Pat_Last	
Player Name - Name Length	 He	

To deal with situations where there are no states that match the current situation in game, you can create a path with a default or fallback state. These “fallback” paths contain one or more state groups instead of states and are usually associated with a more general sound object. In the previous example, the fallback path is associated with the sound object “he” instead of one of the player's names.

After you have re-created all the conditions in the dialogue event, it can be integrated into the game engine. When the dialogue events are called by the game, the sound engine resolves the dialogue event by returning the audio object that corresponds to the matching path. The sound engine can then decide whether to insert the audio object into a dynamic sequence for playback. The relationship between returning an audio object and inserting it into a dynamic sequence does not have to be 1:1. This means that for each resolved dialogue event, a returned audio object can be added to the dynamic sequence as many times, as necessary.



Note

When creating dynamic dialogue, it is important that you work closely with your audio programmer, since many of the functions are only available in the Wwise SDK, including the ability to stitch words or phrases together into sentences.


Since the game engine uses the dialogue event name, you can create the events, integrate them into the game and then build and fine-tune the contents of each event without ever having to re-integrate them into the game again. This gives you a great deal of flexibility to add or remove states, and to experiment with different sounds, all without additional programming.



Caution

Adding, removing, and moving state groups within a dialogue event will automatically modify the paths. Although this is easily done in Wwise, this type of change will require programmer intervention because the code will need to be updated as well.

To help you easily identify a dialogue event in the interface, it is represented by the following icon.

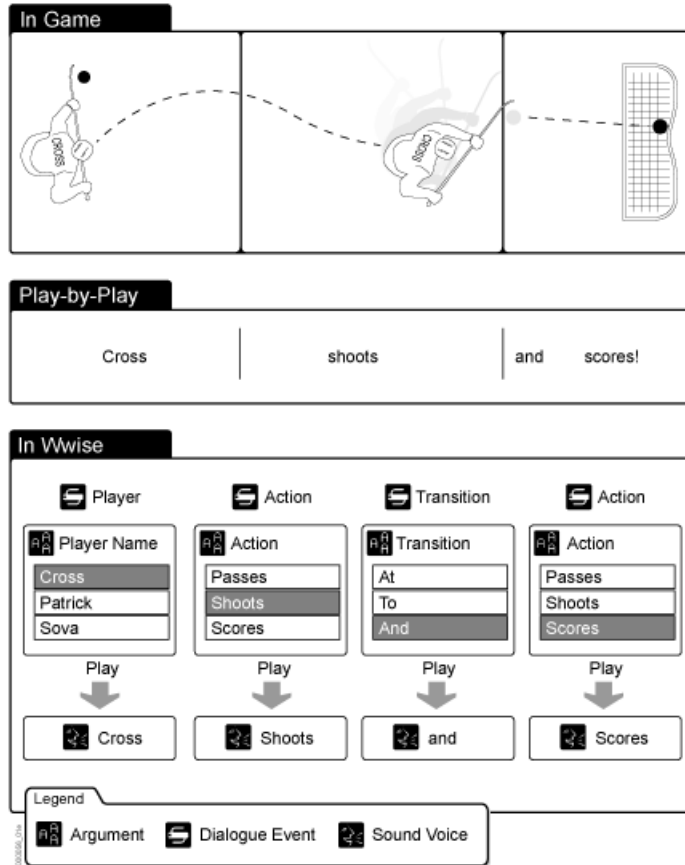
Icon	Represents
	Dialogue event

Using Dialogue Events - Example

Let's say that you are creating a hockey game with a play-by-play commentary. When a player shoots and scores, you want the play-by-play commentary to correspond to the action in game.

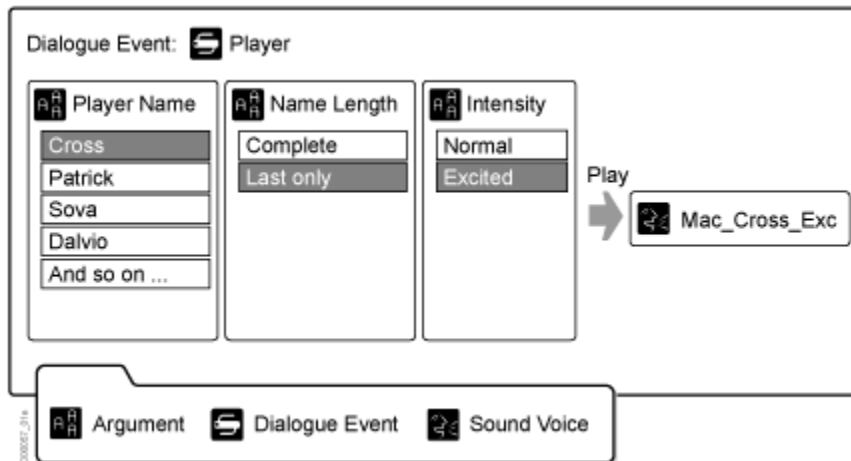
To set up the different possibilities and outcomes in Wwise, you will need to create dialogue events for Players, Actions, Transitions, and so on. Each of these events will contain a set of corresponding state groups and states that you have created for your game. You must create a path that defines each condition or outcome and then assign an appropriate voice object to each path. During gameplay, the game will match the current states against the paths you defined in Wwise to determine which voice object to play.

The following illustration demonstrates how dialogue events created in Wwise can generate a play-by-play commentary that says “Cross shoots and scores”.



The previous illustration was simplified to show you how state groups and dialogue events can be used in your game. In most cases, however, your game will require more sophisticated dialogue events that contain many different state groups and states. Wwise allows you to set up a variety of complex scenarios that can make the dynamic dialogue in your game more realistic.

The following example shows you a more sophisticated example of the Player dialogue event. Notice how this dialogue event has several state groups. When you have several state groups, the selected states create the path. You can create a path for each combination of state groups and states and then assign an object to each path. When each state within a particular path is met in game, the voice object that is assigned to that path will play.



Creating Dialogue Events

The rules for the dynamic dialogue in your game are defined within a dialogue event. The creation of these events involves the following steps:

- [Creating a New Dialogue Event](#)
- [Defining the Settings for a Dialogue Event](#)
- [Adding State Groups to Dialogue Events](#)
- [Re-creating the Conditions in Your Game Using Paths](#)
- [Assigning Objects to Paths](#)

You can manage the dialogue events in the Dialogue Event Editor.



Tip

If you have many paths to create, you can use a shortcut to create them. To speed up the creation process, drag and drop an object from the Project Explorer to the last unselected state in your path. Wwise will automatically create a new path and assign the object to the path. For more information on this shortcut, refer to [Dialogue Events Tips and Best Practices](#).

Creating a New Dialogue Event

At the beginning of your project, it is a very good practice to take the time to define the overall needs of your dynamic audio by analyzing and breaking down the game into manageable components. When you have all the necessary information, you can create a list of all the dialogue events that you will need for your game. After you have created the list, it's much easier to proceed and create them in Wwise.



Note

If you are working as part of a team on the same project, you can assign the dialogue events to different work units so that each member of the team can work on different dialogue events simultaneously. For more information on working with work units, refer to [Dividing Your Project into Work Units](#).

To create a new dialogue event:

1. In the Project Explorer, switch to the **Events** tab.
2. In the Dynamic Dialogue section, do one of the following:

Select a work unit or virtual folder and click the **Dialogue Event** icon in the Project Explorer toolbar.

Right-click a work unit or virtual folder and select **New Child > Dialogue Event** from the shortcut menu.

A new dialogue event is created within the work unit you selected.

3. Replace the default name with one that best represents the dialogue event.



Note

Dialogue event names can contain only letters, numbers, and underscores. They must also start with either a letter or underscore.

Related Topics

- [Adding State Groups to Dialogue Events](#)
- [Defining the Settings for a Dialogue Event](#)
- [Re-creating the Conditions in Your Game Using Paths](#)
- [Creating Paths using States](#)
- [Creating Fallback Paths](#)
- [Assigning Objects to Paths](#)
- [Assigning Probability and Weighting to Paths](#)
- [Deleting Dialogue Events](#)
- [Playing Back Objects within a Dialogue Event](#)

Defining the Settings for a Dialogue Event

For each dialogue event, you need to define the probability and mode settings. The probability setting helps you control the likelihood that the dialogue event

will actually submit an audio object to the sound engine for playback. Once submitted, the sound engine can then decide whether to insert the audio object into a dynamic sequence or not. A value of 100% means that an audio object is always submitted to the sound engine each time the dialogue event is called by the game. A value of 0, on the other hand, means that an audio object is never submitted. This option is particularly useful when you don't want audio to be played back each time the dialogue event is triggered. For example, in a fighting match, you might not want the commentator to say something each time contact is made. To avoid the over-triggering of audio in this type of scenario, you can reduce the probability setting for the dialogue event.

The Mode setting gives you additional control over the behavior of the dialogue event in situations where several paths match the current condition of the game. You can choose between two options: Best Match and Weighted.

To define the settings for a dialogue event:

1. Load a dialogue event into the Dialogue Event Editor.
2. Specify a Probability value for the dialogue event between 0-100. The Probability value determines the likelihood that the dialogue event will submit an audio object to the sound object for playback.



Note

The probability of the dialogue event works in combination with the paths' probability values.

3. To specify how the sound engine will determine which path to select when there is more than one pre-defined path that matches the states triggered at runtime, select one of the following options from the Mode list:

Best Match - Selects the path that best matches the states that are triggered at runtime. If there is not an exact match, the path with the least amount of wildcards (*) will be selected.

Weighted - Randomly selects one of the matching paths based on their respective weighting values.

Related Topics

- [Creating a New Dialogue Event](#)
- [Adding State Groups to Dialogue Events](#)
- [Re-creating the Conditions in Your Game Using Paths](#)
- [Creating Paths using States](#)
- [Creating Fallback Paths](#)

- [Assigning Objects to Paths](#)
- [Assigning Probability and Weighting to Paths](#)
- [Deleting Dialogue Events](#)
- [Playing Back Objects within a Dialogue Event](#)

Adding State Groups to Dialogue Events

Each dialogue event contains a set of rules that reflects the different conditions and outcomes that exist within your game. These rules are defined using state groups and states. When you add a state group to a dialogue event, the corresponding states are also added.

To add state groups to a dialogue event:

1. Load a dialogue event into the Dialogue Event Editor.
2. In the Project Explorer, switch to the Game Syncs tab.
3. In the State Groups section, select one or more state groups that you want to add to the dialogue event.
4. Drag the state groups to the State Groups pane of the Dialogue Event Editor.

The state groups and corresponding states are displayed in the Dialogue Event Editor.

Related Topics

- [Creating a New Dialogue Event](#)
- [Defining the Settings for a Dialogue Event](#)
- [Re-creating the Conditions in Your Game Using Paths](#)
- [Creating Paths using States](#)
- [Creating Fallback Paths](#)
- [Assigning Objects to Paths](#)
- [Assigning Probability and Weighting to Paths](#)
- [Deleting Dialogue Events](#)
- [Playing Back Objects within a Dialogue Event](#)

Re-creating the Conditions in Your Game Using Paths

After the dialogue event is populated with state group and states, you need to define the rules by re-creating each condition and outcome that exists in your game. This is performed by combining the different state groups and states into paths.

Since it may be difficult to cover every situation or outcome, you will most likely want to create fallback or default paths that can be used every time the sound engine encounters a situation where a specific path has not been defined. These fallback paths use state groups instead of states.

The two different types of paths are as follows:

- [Creating Paths using States](#)
- [Creating Fallback Paths](#)

Creating Paths using States

You can re-create the different conditions and outcomes in your game by selecting one state from each of the different state groups in your dialogue event. The selected states create a path. Each path can be associated with a piece of dialogue so that when a specific condition or outcome occurs in game, the appropriate voice object is played.



Tip

If you have many paths to create, you can use a shortcut to create them. To speed up the creation process, drag and drop an object from the Project Explorer to the last unselected state in your path. Wwise will automatically create a new path and assign the object to the path. For more information on this shortcut, refer to [Dialogue Events Tips and Best Practices](#).

To create a path:

1. Load a dialogue event into the **Dialogue Event Editor**.
2. Click a state for each of the state groups in your dialogue event.

Add Path becomes active.

3. Click **Add Path** to create a path.

The path is added the path list.

4. Continue to add new paths by selecting a different combination of states.



Note

To remove a path, select the series of states or the path itself and click **Remove Path**.

Related Topics

- [Creating a New Dialogue Event](#)
- [Defining the Settings for a Dialogue Event](#)
- [Adding State Groups to Dialogue Events](#)
- [Creating Fallback Paths](#)

- [Assigning Objects to Paths](#)
- [Assigning Probability and Weighting to Paths](#)
- [Deleting Dialogue Events](#)
- [Playing Back Objects within a Dialogue Event](#)

Creating Fallback Paths

Since it can be difficult, time-consuming, and very expensive to re-create every condition and outcome that exists in a game, you will want some kind of fallback mechanism to cover those situations you didn't think of or didn't have time to create. In Wwise, this is done by creating fallback paths. These fallback paths contain one or more state groups instead of states and are generally associated with more general pieces of dialogue. For example, in an RTS game, you might have situations where the outcome of a certain operation is uncertain, such as the final destination of your army. To cover this situation, you can create a path where at least one state group is used in place of a state. This path would then be assigned to a sound object that leaves the operation open-ended. In the case of your army on the way to its final destination, the sound object might be “Your army is advancing”.

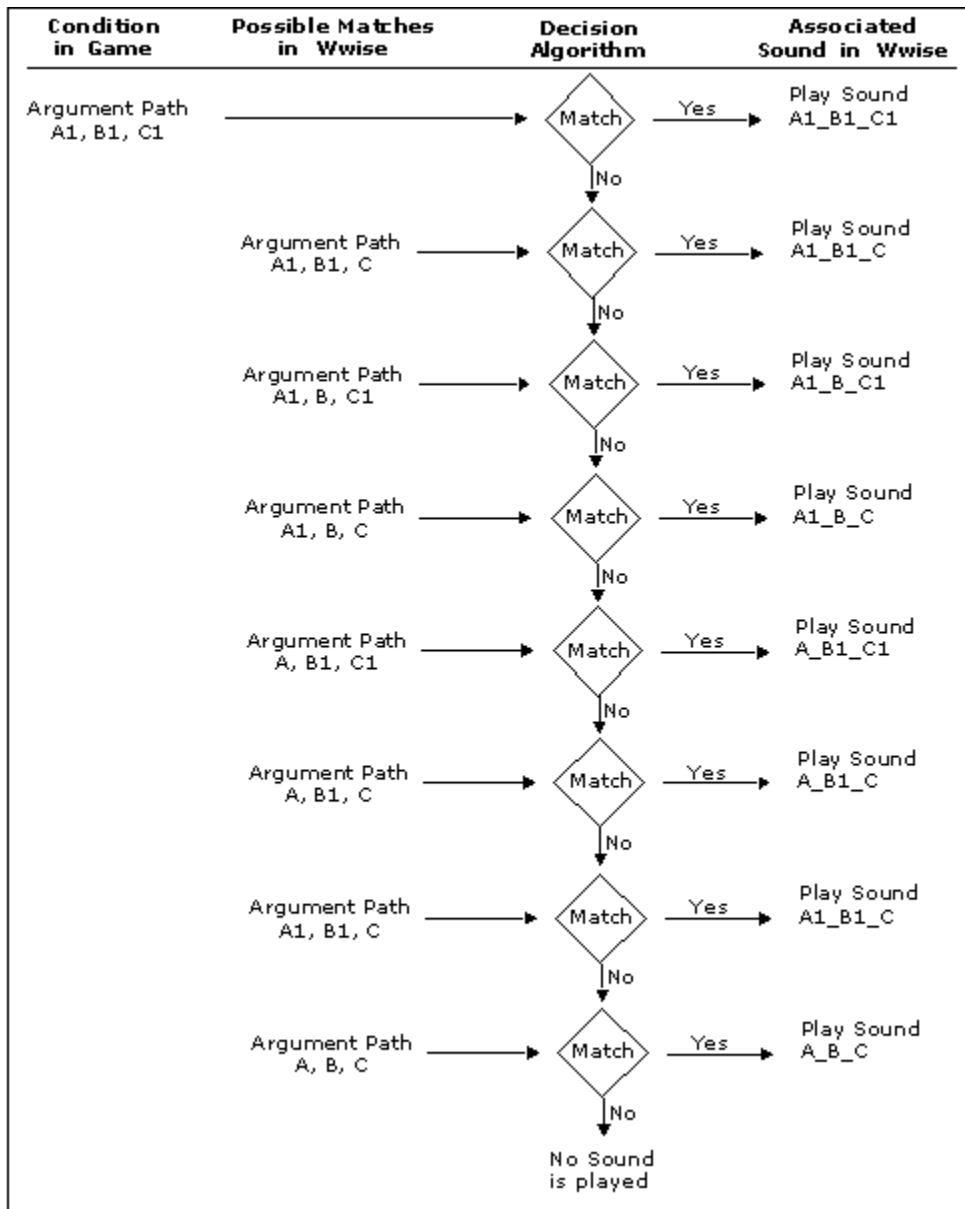
Before creating the dialogue for a game, it is important that the scriptwriter understand how the fallback algorithm works in Wwise. Let's say you have a dialogue event with the following state groups and states:

Arguments	A	B	C
	A1	B1	C1
Argument paths	A2	B2	C2
	A3	B3	C3

When the dialogue event is called by the game, the algorithm will try to match the current condition in game, let's say (A1, B1, C1), with one of the paths in the dialogue event to determine which sound to play. When more than one path matches the current condition in game, the mode of the dialogue event will ultimately decide how Wwise picks a path and corresponding sound. When a dialogue event is in 'Best Match' mode, Wwise will look for an exact match by verifying the current condition in game against the paths created in the dialogue event. If no exact match is found or if the matching path doesn't have an associated piece of dialogue, the algorithm will look for the closest match among the fallback paths, if any.

When a dialogue event is in 'Weighted' mode, Wwise looks for all matches including all fallback paths. From this list of matches, it looks at the weighting of each path and from there determines which path to select and which piece of dialogue to play.

The following illustration shows how the 'best match' decision algorithm verifies all possible matches to determine which object to play.



Notice how the decision algorithm looks for fallbacks from right to left. In our example, it starts with C, then looks at B, and ends at A. The left-most fallback will only be considered when all other possible combinations are exhausted.



Note

You can also create a generic path for a dialogue event. The generic path covers all situations and ensures that a piece of

dialogue will be played no matter which states are triggered by the game.

To create a fallback path:

1. Load a dialogue event into the Dialogue Event Editor.
2. Click a state group or state for each of the state groups in your dialogue event.

The Add Path button becomes active.



Note

At least one state group needs to be selected in your path to create a fallback path.

3. Click **Add Path** to create a path.

The path is added the path list.



Note

When a state group is used to create a fallback path, the state group is represented by an asterisk(*) in the path name.

4. Continue to add new paths by selecting a different combination of state groups and states.



Note

To remove a path, select the series of states or the path itself and click Remove Path.

Related Topics

- [Creating a New Dialogue Event](#)
- [Defining the Settings for a Dialogue Event](#)
- [Adding State Groups to Dialogue Events](#)
- [Creating Paths using States](#)
- [Assigning Objects to Paths](#)
- [Assigning Probability and Weighting to Paths](#)

- [Deleting Dialogue Events](#)
- [Playing Back Objects within a Dialogue Event](#)

Assigning Objects to Paths

After defining the paths in your dialogue event, you need to associate each path with an object. By assigning an object to a path, you specify which object will play when all the states within the path are present in game. To add variety to each condition, you can assign a container to a path as well.

At any point, you can change the path that is assigned to a particular object by simply selecting the object in the list, picking a new path in the State Groups pane, and then clicking the Update Path button. This button will only be available when the newly selected path in the State Groups pane does not already have a corresponding entry in the list.

To help you identify the status of objects assigned to a path, the object name will appear in one of the following colors:

- **White** - For objects included in the current platform.
- **Gray** - For objects not included in the current platform.
- **Red** - For objects that are missing from the current project.
- **Yellow** - For objects that are currently unloaded from the current project.



Tip

You can create a path and assign an object to that path in one step. When creating the path, simply drag and drop an object from the Project Explorer to the last unselected state in your path. Wwise will automatically create a new path and assign the object to the path. For more information on this shortcut, refer to [Dialogue Events Tips and Best Practices](#).

To assign objects to paths:

1. Load a dialogue event into the Dialogue Event Editor.
2. In the path list, click the Browse button (...) that corresponds to the path to which you want to assign an object.

The Project Explorer - Browser opens.

3. Select the container or object that you want to assign to the path.
4. Click OK.

The object is assigned to the path and is displayed in the Object column.



Tip

To remove an object from a path, right-click the path and select **Set to none** from the menu.

Related Topics

- [Copying Settings from One Platform to Another](#)
- [Creating a New Dialogue Event](#)
- [Defining the Settings for a Dialogue Event](#)
- [Adding State Groups to Dialogue Events](#)
- [Re-creating the Conditions in Your Game Using Paths](#)
- [Creating Paths using States](#)
- [Creating Fallback Paths](#)
- [Assigning Probability and Weighting to Paths](#)
- [Filtering the Path List](#)
- [Deleting Dialogue Events](#)
- [Playing Back Objects within a Dialogue Event](#)

Filtering the Path List

If you are dealing with many state groups and states within the same dialogue event, the path list can become very full. If the list becomes unmanageable, you can always filter it using one of the filter options.



Note

You can also sort the path list alphabetically by clicking the Path title bar.

To filter the path list:

1. Load a dialogue event into the Dialogue Event Editor.
2. From the Path Filter list, select one of the following options:

All to display all the created paths.

Current Selection to display only those paths that contain the selected state groups or states.

3. From the Filter list, select one of the following options:

All to display all the created paths.

Assigned object to display only the paths that are associated with an object.

Missing to display only the paths that are associated with an object that has been deleted from the project.

None to display only the paths that are not associated with an object.

Related Topics

- [Creating a New Dialogue Event](#)
- [Defining the Settings for a Dialogue Event](#)
- [Adding State Groups to Dialogue Events](#)
- [Re-creating the Conditions in Your Game Using Paths](#)
- [Creating Paths using States](#)
- [Creating Fallback Paths](#)
- [Assigning Objects to Paths](#)
- [Assigning Probability and Weighting to Paths](#)
- [Deleting Dialogue Events](#)
- [Playing Back Objects within a Dialogue Event](#)

Assigning Probability and Weighting to Paths

Wwise gives you additional tools to manage the selection and playback of paths within your game. For each path within a dialogue event, you can assign Probability and Weight values. When there are several matching paths, these properties define the likelihood that a path will be selected and the corresponding audio object will be played back.

To assign probability and weighting to paths:

1. Load a dialogue event into the Dialogue Event Editor.
2. Use the **Probability** slider to assign a probability to the path.

The probability value determines the likelihood that when a particular path is selected, it will result in audio being played back. The final playback probability is the combination of the probability of the selected path and the dialogue event.

3. Use the **Weight** slider to assign a weight to the path.

The weight value helps to prioritize certain paths over others. Paths with a higher weight are more likely to be selected in the case where there are other matching paths. When a matching path has a weight of 100, other matching paths with a weight below 100 are automatically discarded. When a matching path has a weight of 0, it is discarded unless all other matching paths also have a weight of 0.



Note

The Weight option is only available when the dialogue event is in 'Weighted' mode.

Related Topics

- [Creating a New Dialogue Event](#)
- [Defining the Settings for a Dialogue Event](#)
- [Adding State Groups to Dialogue Events](#)
- [Re-creating the Conditions in Your Game Using Paths](#)
- [Creating Paths using States](#)
- [Creating Fallback Paths](#)
- [Filtering the Path List](#)
- [Assigning Objects to Paths](#)
- [Deleting Dialogue Events](#)
- [Playing Back Objects within a Dialogue Event](#)

Working with Dialogue Events

If you plan to use dialogue events to drive the audio in your game, it is important that you meet with the development team at the beginning of the project to define all the different conditions and outcomes that will exist in your game.

Since the paths are a set of rules that are verified when the dialogue event is called by the game, any change that is made to the paths, including the number and order of state groups, will result in additional programming. By taking the time up front to define all the different rules, variables, and conditions, you can avoid problems and minimize the need for additional programming.

When managing dialogue events, you can perform the following tasks:

- [Renaming a Dialogue Event](#)
- [Re-ordering State Groups in a Dialogue Event](#)
- [Removing State Groups from a Dialogue Event](#)
- [Deleting Dialogue Events](#)
- [Playing Back Objects within a Dialogue Event](#)

Renaming a Dialogue Event

Wwise automatically gives a name to a dialogue event when it is created. It is good practice to rename the dialogue event with a name that is more

descriptive. Each event name must be unique, and consist only of letters, digits, and underscores. The first character must be either a letter or an underscore.



Caution

Unless absolutely necessary, you shouldn't rename a dialogue event after it has been integrated into the game because it will require additional programming to re-integrate the name change.

To rename a dialogue event:

1. In the Events tab of the Project Explorer, click the dialogue event you want to rename.

The name of the event is highlighted.

2. Type a new name for the event.



Tip

You can also rename events in the Dialogue Event Editor.

Related Topics

- [Creating a New Dialogue Event](#)
- [Adding State Groups to Dialogue Events](#)
- [Assigning Objects to Paths](#)
- [Re-ordering State Groups in a Dialogue Event](#)
- [Removing State Groups from a Dialogue Event](#)
- [Deleting Dialogue Events](#)
- [Playing Back Objects within a Dialogue Event](#)

Re-ordering State Groups in a Dialogue Event

The order of the state groups within a dialogue event is important because it defines the sequence of values within each path. If you placed the state groups in the wrong order, you can re-order them at any time.



Caution

Unless absolutely necessary, you shouldn't re-order the state groups within a dialogue event after it has been integrated into

the game because it will require additional programming. If you must re-order the state groups, make sure to talk to your audio programmer so that he can make the corresponding changes in the code.

To re-order the state groups in a dialogue event:

1. Load a dialogue event into the Dialogue Event Editor.
2. In the State Groups pane of the Dialogue Event Editor, drag a state group to the new location. A red line appears to help you position the state group.

The existing paths are automatically updated to reflect the new order of the state groups.

Related Topics

- [Creating a New Dialogue Event](#)
- [Adding State Groups to Dialogue Events](#)
- [Assigning Objects to Paths](#)
- [Renaming a Dialogue Event](#)
- [Removing State Groups from a Dialogue Event](#)
- [Deleting Dialogue Events](#)
- [Playing Back Objects within a Dialogue Event](#)

Removing State Groups from a Dialogue Event

During the development of your project, you can add or remove one or more state groups to your dialogue events.



Caution

Unless absolutely necessary, you shouldn't add or remove state groups from dialogue event after it has been integrated into the game because it will require additional programming. If you must add or remove a state group, make sure to talk to your audio programmer so that he can make the corresponding changes in the code.

To remove state groups from a dialogue event:

1. Load a dialogue event into the Dialogue Event Editor.
2. Right-click the state group that you want to remove and select **Remove Column** from the shortcut menu.

The state group and its states are removed from the dialogue event.

Related Topics

- [Creating a New Dialogue Event](#)
- [Adding State Groups to Dialogue Events](#)
- [Assigning Objects to Paths](#)
- [Renaming a Dialogue Event](#)
- [Re-ordering State Groups in a Dialogue Event](#)
- [Deleting Dialogue Events](#)
- [Playing Back Objects within a Dialogue Event](#)

Deleting Dialogue Events

If you no longer need a dialogue event, you can delete it. Before deleting a dialogue event, you may want to verify if it is being used by another member of your team in different part of the project or if it is already included in one of your SoundBanks. If you or someone else on your team deletes a dialogue event that is included in a SoundBank, it will create an invalid dialogue event. Wwise doesn't automatically remove deleted dialogue events from SoundBanks, so you will have to remove them manually. To help you track and manage these types of dialogue events, Wwise displays a complete list in the SoundBank Manager. For more information on deleting invalid events, refer to [Searching for Elements within a SoundBank](#). You can also view a complete list of invalid dialogue events in the project integrity report. For more information on using the integrity report, refer to [Troubleshooting Your Project](#).

To delete an event:

1. On the Events tab of the Project Explorer, right-click the dialogue event you want to delete and select **Delete Selection** from the shortcut menu.

The dialogue event is deleted.



Note

If you delete a dialogue event by mistake, you can undo the delete, by pressing **Ctrl+Z** or by clicking **Edit > Undo**.

Related Topics

- [Creating a New Dialogue Event](#)
- [Adding State Groups to Dialogue Events](#)

- [Assigning Objects to Paths](#)
- [Renaming a Dialogue Event](#)
- [Re-ordering State Groups in a Dialogue Event](#)
- [Removing State Groups from a Dialogue Event](#)
- [Playing Back Objects within a Dialogue Event](#)

Playing Back Objects within a Dialogue Event

At any point in the creation process, you can audition the object assigned to any one of the paths within a dialogue event.

To play back a dialogue event:

1. Load a dialogue event into the Dialogue Event Editor.
2. Select a path.

The assigned sound object is automatically loaded into the Transport Control.

3. Click the Play icon in the Transport Control.

The sound object is played back.

Related Topics

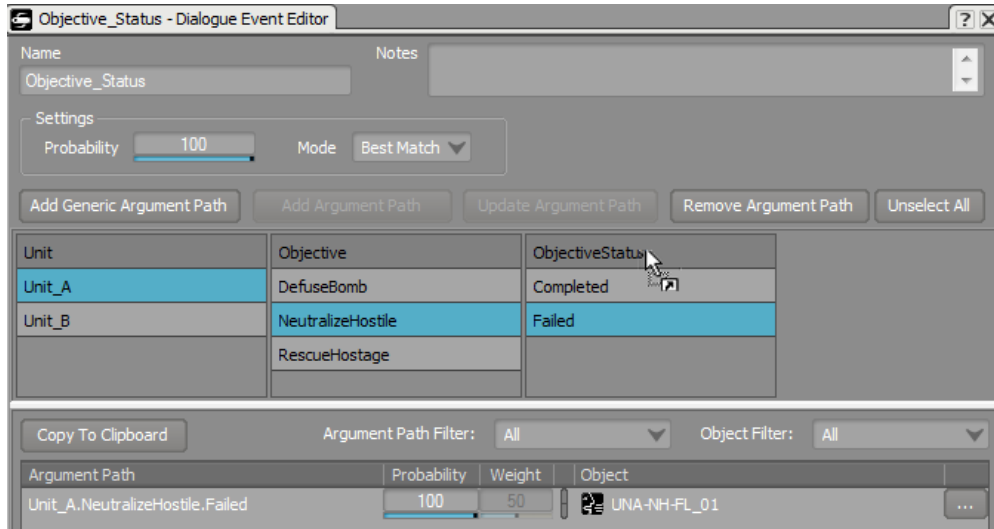
- [Creating a New Dialogue Event](#)
- [Adding State Groups to Dialogue Events](#)
- [Assigning Objects to Paths](#)
- [Renaming a Dialogue Event](#)
- [Re-ordering State Groups in a Dialogue Event](#)
- [Removing State Groups from a Dialogue Event](#)
- [Deleting Dialogue Events](#)

Dialogue Events Tips and Best Practices

Before creating dialogue events, you may want to review the following sections, which provide you with a series of tips and best practices that can help you efficiently create and manage the dialogue events in your project.

Shortcut for Creating Paths

If you have many paths to create, you can use a shortcut to create them. This shortcut allows you to finalize the path and associate a sound object at the same time. Before finalizing the contents of your path, drag the corresponding object from the Project Explorer to the last unselected state group or state.



Wwise will automatically create a new path and assign the object to it. If a path already exists, the currently assigned object will be replaced with the new one.

Strategies for Building Paths

- Create fallback paths first - as a general rule, it is a good idea to create all the fallback paths for a dialogue event first to make sure that you've covered all conditions and outcomes. After these more general fallback paths are created, you can then start building the more specific paths. If you don't have time to re-create every condition or if new conditions are added late in the development cycle, you are already covered.
- Order state groups to maximize algorithm efficiency - state groups that are less likely to use a fallback should be placed first in the order of state groups. This reduces the amount of backtracking the algorithm must do to resolve a path, which improves the overall efficiency of the dynamic dialogue system.

In the following example, the Unit state group was placed first because it will not have a fallback. A fallback for this state group wouldn't make sense because Unit A and Unit B are associated with specific characters in the game making it impossible to create a generic “person”. Since it makes sense to create fallbacks for the Objective and ObjectiveStatus state groups, they should come after the Unit state group.

The screenshot shows the Dialogue Event Editor interface. At the top, there is a title bar "Dialogue Event Editor". Below it, there are fields for "Name" (containing "Objective_Status") and "Notes". A "Settings" section contains a "Probability" slider set to 100 and a "Mode" dropdown menu set to "Best Match". Below the settings are four buttons: "Add Generic Argument Path", "Add Argument Path", "Update Argument Path", and "Remove Argument Path".

Unit	Objective	ObjectiveStatus
Unit_A	DefuseBomb	Completed
Unit_B	NeutralizeHostile	Failed
	RescueHostage	

Annotations below the table:

- Under the Unit column: No fallback for this argument
- Under the Objective column: Fallback for this argument
- Under the ObjectiveStatus column: Fallback for this argument

Chapter 16. Working with States

Overview	408
Working with States	409
Defining Transitions Between States in a State Group	413
Assigning States to Objects and Busses	414
States Tips and Best Practices	420

Overview

Game designers are constantly challenged to create the most compelling audio possible using the least amount of memory, CPU, assets, and disk space. States provide an efficient and creative way to approach this challenge. Using states optimizes your sound and music assets by giving you the flexibility to create different 'mixer snapshots' for the same sound, and apply these property changes globally in response to changes in the game. By changing the properties of a sound or music object, you can creatively match the action in the game without adding new assets. When you are planning your project, you can decide when and where states will have the most efficient and creative impact.



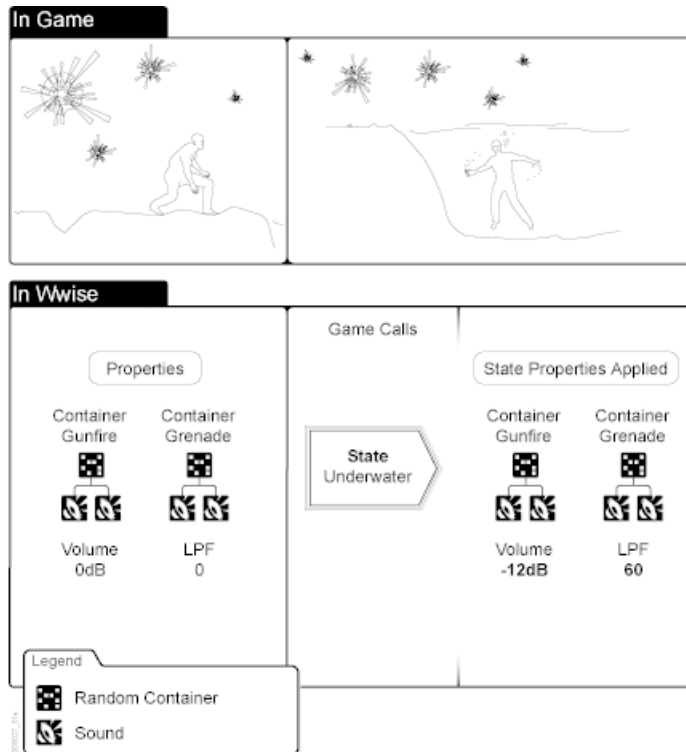
Note

States can also be used to efficiently modify the properties of motion FX objects in your project. The principles and workflow are the same for motion as they are for sounds, except that the Low Pass Filter properties have no effect on motion objects.

Using States - Example

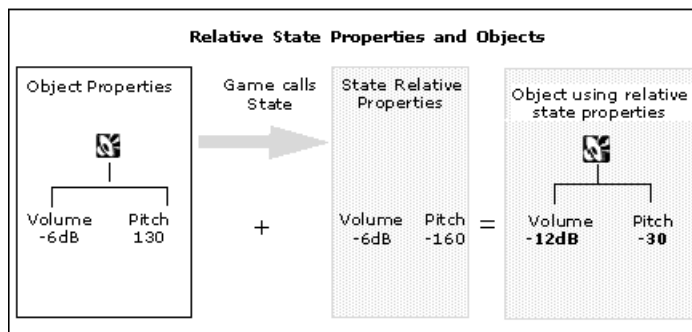
Let's say you want to simulate the sound treatment that occurs when a character goes underwater. In this case you could use a state to modify the volume and low pass filter for sounds that are already playing. These property changes should create the sound shift needed to recreate how gunfire or exploding grenades would sound when the character is under water.

The following illustration demonstrates how the properties for the volume and low pass filter for the gunfire and grenade sound objects are affected when the underwater state is called by the game.



Understanding How State Properties are Applied to Objects

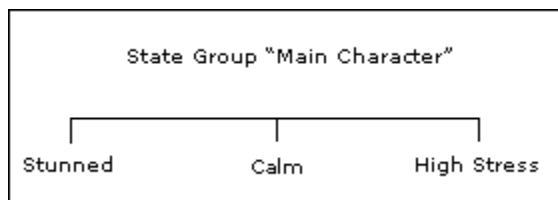
State properties are always relative. When you apply a state, the effect on the object's properties will be cumulative.



Working with States



A state needs to belong to a state group before it can be applied to any object in Wwise. You can logically organize states into state groups in ways that make it easier to manage them. For example, you would probably find it useful to group together states associated with the main character in a game. You would then create a state group named Main Character, to which you can add states that will be applied to the properties for the objects associated with the Main Character. From the game, you know that the main character will probably

experience the following states - stunned, calm, high stress. So it would be useful to group these together and then define the property changes associated with the individual states.



After you have set up your state structure, you can subscribe objects to state groups in the Property Editor and customize the state properties as needed.

To help you easily identify a state group or a state in the interface, Wwise uses a unique icon to identify them.

Icon	Represents
	State group
	State

Working with states can include the following tasks:

- [Creating a State Group](#)
- [Creating a State](#)
- [Defining Transitions Between States in a State Group](#)
- [Deleting States/State Groups](#)

Creating a State Group

You can create all the state groups that you need from either of the following two places in Wwise:

- [Project Explorer](#)
- [States tab of the Property Editor](#)

To create a new state group for your project from the Project Explorer:

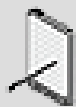
1. In the Project Explorer, switch to the **Game Syncs** tab.
2. In the States section, do one of the following:

Select a work unit or virtual folder and click the **State Group** icon in the Project Explorer toolbar.

Right-click a work unit or virtual folder and click **New Child > State Group** from the shortcut menu.

A new state group is added to the list of state groups.

3. Replace the default name with one that best represents the state group.



Note

Each state group name must be unique, and consist only of letters, digits, and underscores. Use either a letter or underscore as the first character.

4. Continue to add state groups as needed.

To create a new state group for your project from the States tab of the Property Editor:

1. In the States tab of the Property Editor, click **Add >>**.

The State selector menu opens.

2. Select **New...**
3. Select the work unit into which you want to create a new state group.
4. In the Name field, replace the default name with one that best represents the state group.



Note

Each state group name must be unique and consist only of letters, digits, and underscores. Use either a letter or underscore as the first character.

5. Click **OK** to create and subscribe to the new state group.

Related Topics

- [Creating a State](#)
- [Defining Transitions Between States in a State Group](#)
- [Deleting States/State Groups](#)

Creating a State

When a state is called by the game, it applies property changes to objects in response to the game conditions. The transitions between the states are defined in the State Group editor.

The process of creating a new state includes the following steps:

- [Creating a State](#)

To create a new state:

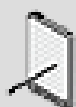
1. In the Project Explorer, switch to the **Game Syncs** tab.
2. In the States section, do one of the following:

Select a state group and click the **State** icon in the Project Explorer toolbar.

Right-click a state group and select **New Child > State** from the shortcut menu.

A new state is added to the state group.

3. Replace the default name with one that best represents the state.



Note

Each state name in a state group must be unique, and consist only of letters, digits, and underscores. Use either a letter or underscore as the first character.

4. Continue creating states as needed.

Related Topics

- [Creating a State Group](#)
- [Defining Transitions Between States in a State Group](#)
- [Deleting States/State Groups](#)

Deleting States/State Groups

You may need to delete a state or a state group that you no longer need in your project. Keep in mind that deleting a state group also deletes all the states in that state group. Deleted states will no longer be available to the objects and presets that were using them.



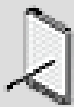
Tip

Use the **Find All References** command on the State Group's shortcut menu to find which object is using the State Group before you delete it.

To delete a state group:

1. In the Project Explorer, switch to the **Game Syncs** tab.
2. In the States section, right-click the state group or state that you want to delete, and select **Delete Selection**.

The selected state or state group is deleted.



Note

If you delete a state or state group by mistake, you can undo the delete by pressing Ctrl+Z, or by clicking Edit > Undo.

Related Topics

- [Creating a State Group](#)
- [Creating a State](#)

Defining Transitions Between States in a State Group

To provide smooth transitions between two states within the same state group, you can define the time between state changes. You have two options when defining the transition time:

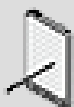
- [Defining Transitions for all States in a State Group](#) to set the same transition time between states for all states in a state group.
- [Customizing Transitions between States in a State Group](#) to define different transition times between states in a state group.

Defining Transitions for all States in a State Group

You can define a single transition time that will be used between all states within a state group.

To define the transition time between states in a selected state group:

1. In the Project Explorer, switch to the **Game Syncs** tab.
2. In the States section, double-click a state group to load it into the Property Editor.
3. In the default transition time field, define a transition time for all the states in the selected state group.



Note

The default transition time is used when no custom transition time has been defined.

Related Topics

- [Creating a State](#)

- [Deleting States/State Groups](#)
- [Customizing Transitions between States in a State Group](#)

Customizing Transitions between States in a State Group

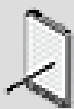
To enhance realism as states change within your game, you may not want the transition time to be the same for all states in a state group. In the Custom Transition Time table, you can define different transition times between any two states. You can also specify that you want a customized transition time to be the same in both directions. For example, you may want to use the same transition time when the Rain state changes to the Snow state and vice versa.

To customize the transition time between states in a selected state group:

1. In the Project Explorer, switch to the **Game Syncs** tab.
2. In the States section, double-click a state group to load it into the Property Editor.
3. Click **Insert**.

A new row is added to the Transition Time table.

4. In the From column, select the source state.
5. In the Time column, define the transition time that you want to apply between the two states.
6. In the To column, select the destination state.
7. To have the same transition time in both directions, select the bidirectional check box.



Note

You can add more custom settings rows by clicking **Insert**, and if you want to remove settings, click **Remove**.

Related Topics

- [Creating a State](#)
- [Deleting States/State Groups](#)
- [Defining Transitions for all States in a State Group](#)

Assigning States to Objects and Busses

After you have created and defined the states for your project, you are ready to assign state groups and states to objects and busses so that the sounds, music,

and motion can match what is going on in game. You can then make specific adjustments to the properties to further differentiate the sound, music, or motion.

An object must first subscribe to one or multiple state groups, and then all states within those state groups are automatically assigned to the object. Each object can subscribe to multiple state groups. You can also assign state groups at each level in the Master-Mixer, Actor-Mixer, or Interactive Music hierarchies.

To assign a State Group to an object:

1. Load an object into the Property Editor.
2. Switch to the **States** tab.
3. Click the **Add >>** button and select the state group that you want to assign to the object.

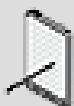
The associated states appear in the list. The object is now subscribed to the states in the selected state group.

Related Topics

- [Customizing State Properties for Objects](#)
- [Defining Points within Your Music Objects for State Changes](#)

Defining Points within Your Music Objects for State Changes

In interactive music it is very useful to time your state changes so they accommodate the tempo of the music that is playing and ensure smooth transitions. This can be defined at the music object property level and at the bus level. In the Property Editors for music objects or busses, you can determine the optimal points to change states for each of the subscribed state groups. These points can include Immediate, Next Cue, or Next Beat, for example. When however, a state change involves multiple music objects with different settings for the state changes, the state change will occur at the next defined opportunity in the segment.



Note

If an audio bus only contains **Actor-Mixer Hierarchy** sound objects, these settings will be ignored and the changes will occur immediately. If however, both music and sound objects are routed through an audio bus, the state change will be based on the state change settings of the music objects.

To define a point within music objects and busses for a state change:

1. Load a music object or bus into the Property Editor.
2. Switch to the States tab.
3. For one of the subscribed state groups, select one of the following options in the Change occurs at column:

Immediate - Change occurs immediately. If a look ahead time has been defined for a track, then this time must elapse before the state change can occur.

Next Grid - Change occurs at next grid. The grid is an arbitrary frequency by which music objects can be virtually partitioned.

Next Bar - Change occurs at next bar.

Next Beat - Change occurs at next beat.

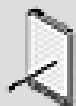
Next Cue - Change occurs at next cue. The next cue could be an Entry, Exit, or custom cue.

Next Custom Cue - Change occurs at next custom cue.

Entry Cue - Change occurs at Entry cue.

Exit Cue - Change occurs at Exit cue.

All state changes in the current state group will occur at the specified point.



Note

If several music objects subscribe to a state group, a state change is applied only once, for all objects at the same time. The state change will occur for all the objects at the first defined opportunity.

Related Topics

- [Customizing State Properties for Objects](#)

Customizing State Properties for Objects

After you have assigned a state group to an object, you can customize the properties of the individual states for that object. For each state, you can modify the following properties:

- Pitch
- Low-Pass Filter
- High-Pass Filter
- Volume



Note

Not all properties are used by all objects and on all platforms. For example, the **Pitch** property does not affect music objects and the **LPF** property does not affect motion objects.

To customize the state properties for an object:

1. Load an object into the Property Editor.
2. Switch to the **States** tab.
3. Set the values for the following properties:

Pitch to increase or decrease the playback speed of the sound or motion object.

Low Pass Filter to apply a recursive filter that attenuates high frequencies.

High Pass Filter to apply a recursive filter that attenuates low frequencies.

Volume to adjust the level or amplitude of the output.

Related Topics

- [Assigning States to Objects and Busses](#)
- [Copying State Values Between States](#)
- [Example: Creating a Temporary Loss of Hearing Effect](#)

Copying State Values Between States

If you have many states that use similar property settings, you can define the settings once and then copy these state values to the other states within the same state group.

You can copy the values of a state to an existing or newly-created state within the same state group. You can even decide which objects will be affected by the source's state values.

To copy state values:

1. Click the **Copy State Values** button from one of the following views:

State Property Editor.

Mixing Desk.

The **Copy States Values** dialog box opens.



Tip

You can also right-click a state and select **Copy State Values** from either the **States** tab of the **Property Editor** or from within the **Mixing Desk**.

2. Click the **State Group Selector** button (>>) and select the state group that contains the state whose custom properties you want to copy.
3. Click the **From Selector** button (>>) and select the state whose custom properties you want to copy.

All objects that subscribe to the state group are displayed in the **Affected objects list**.



Note

If you open the **Copy Custom States** dialog using the right-click menu, only the selected object will be displayed in the **Affected objects list**.

4. Click the **To Selector** button (>>) and then do one of the following:

To copy the custom properties to a new state, select **New**, specify a name for the new state, and then click **OK**.

To copy the custom properties to an existing state, simply select the state from the list.

Wwise determines what action to take for each object in the **Affected objects list**.

5. In the **Use** column, select the check box for each object that you want to use the new custom state settings.
6. Click **OK** to apply the custom state settings to the selected objects.

Related Topics

- [Assigning States to Objects and Busses](#)
- [Customizing State Properties for Objects](#)

- [Example: Creating a Temporary Loss of Hearing Effect](#)

Example: Creating a Temporary Loss of Hearing Effect

Let's say you are creating a first-person shooter game where the player experiences a temporary loss of hearing when a flash-bang grenade goes off nearby. To imitate this type of effect, you will need to temporarily affect all sounds.

Steps

The best way to achieve the "temporary loss of hearing" effect is by using states. For example, you can create a **State Group** "GrenadeFX" that consists of two states: "Stunned" and "Normal". For the "Stunned" state, you can modify the **Pitch**, **Volume**, **LFE Volume**, and **Low Pass Filter** values to create your "temporary loss of hearing" effect. This **State Group** can then be used on one or several control busses to affect your SFX, ambiences, music, and so on.

To add more realism, on the same control busses that use the "GrenadeFX"# **State Group**, apply RTPCs to influence the intensity of the proximity effect. The properties of the state (Volume, Pitch, LFE, and LPF) and the RTPC are added so that you can reduce or increase their values based on the proximity of the grenade from the main character/microphone. To achieve this effect, you can do the following:

1. Create a new game parameter in the **Game Syncs** tab, such as "Grenade Proximity"#, and define a Min and Max distance that fits your game.
2. On the control busses where you are using the **State Group** "GrenadeFX"#, go on the **RTPC** tab and assign one or several properties to the game parameter "Grenade Proximity"#.
3. Map the values in such a way that the closer the grenade impact is to the character the more intense the effect will be.

Fine-tune the state and RTPC values directly in the **Soundcaster** as you recreate an in-game simulation. For example, play music, ambient sounds, voice, and SFX, and then switch the **State Group** from "Normal" to "Stunned". Don't forget to fine-tune the **Transition Time** between the different states.

Another way to add more realism is if you muffle the reverb tail effect when your character is in the Stunned state. By using RTPC on the reverb "LPF Cutoff Frequency" property and syncing it with the State changes, you will be able to match your reverb with the amount of LPF you set for the objects.

Related Topics

- [Assigning States to Objects and Busses](#)
- [Customizing State Properties for Objects](#)

States Tips and Best Practices

When working with states, you may want to review the following sections, which provide you with a series of tips and best practices that can help you better manage how you work with states throughout the development process.

Deleting States/State Groups

States are integrated in the game using one of two mechanisms. They can be integrated by calling an event with a set state action, or they can be integrated by calling the state group and the state itself. In the former case, deleting a state or state group will create problems in Wwise as the called state will no longer be available. In the latter case, the sound designer who intends to delete the state or state group should advise the audio programmer of the change.

Use the **Find all References** command to find out where the state is being used in the project.

State Properties and CPU and Memory Usage

In Wwise, certain relative properties such as pitch can affect performance on the different platforms. The mechanism for managing pitch in Wwise is based on the sample rate. Applying pitch to objects increases CPU usage because the files must be resampled.

Chapter 17. Working with Switches

Overview	422
Working with Switches	423
Mapping Game Parameter Values to Switches	426
Switches Tips and Best Practices	427

Overview

In addition to organizing objects in the hierarchy, Wwise also helps you streamline the organization of your sounds, music, and motion objects using switches. A switch represents each alternative that exists for a particular element in game, and is used to help manage the corresponding objects for these alternatives. The alternatives that exist for these game elements are as varied as, for example, the weapon arsenal used by the main character, or weather conditions. You would assign the alternative sounds or motion objects to a certain switch and these objects will play based on the switch that is active in the game.

The following list includes only a few of the many possible game situations or elements where switches can simplify the task of managing the alternatives at runtime:

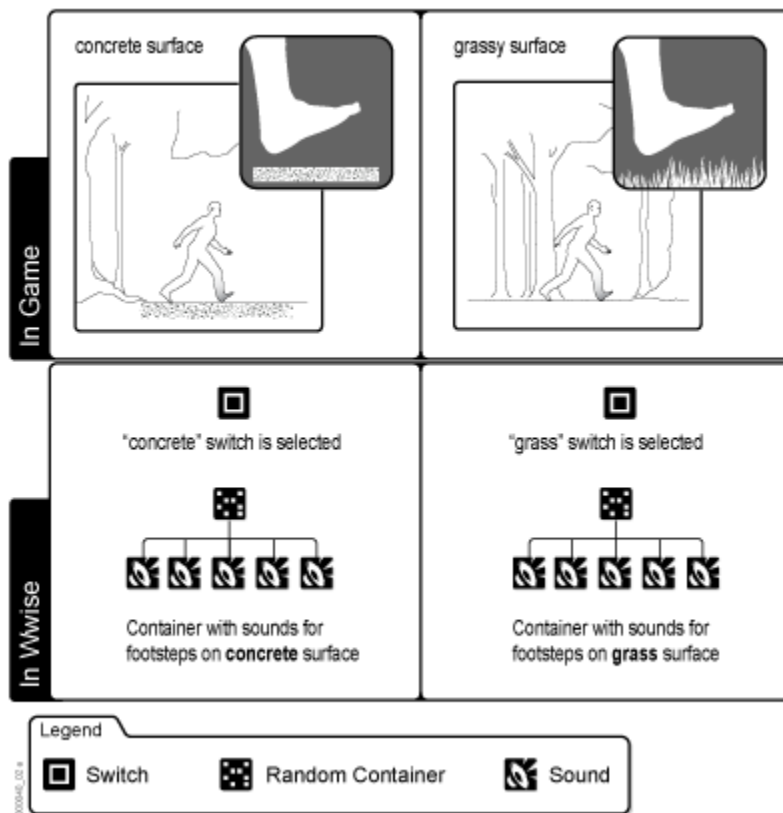
- Game settings for rooms, ground surfaces, indoors/outdoors - you could create switches for different ground surfaces like wood, grass, gravel, and so on.
- Game characters - you could create switches for the dialogue when a male or female character is speaking.
- Weather conditions - you can create switches for tempests, snow storms, gentle rain, or a sunny day.
- Game atmospheres for evil or fairy worlds - you can create switches for the different sounds and motion associated with each world.
- Weapons - you can create switches for the different firing patterns of firearms in your game as well as lasers and swords.

For each of these examples you would create the switch and then assign the corresponding sound, music, or motion objects. The objects that are assigned to a switch are grouped into a switch container. When an event or a game parameter value signals a change, the switch container verifies the switch and the correct object is played.

Using Switches - Example

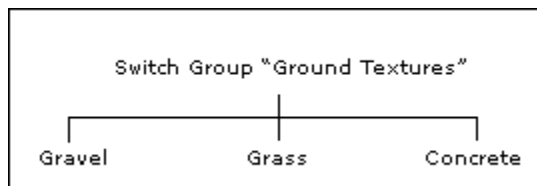
Let's say you are creating a first person shooter game where the character can walk through a variety of different environments. Within each physical setting, you have different ground surfaces, such as concrete, grass, and dirt, which means that you will need different footstep sounds for each of these surfaces. In this case, you can create switches for the different ground surfaces and then assign the different footstep sounds to the appropriate switch. When the main character is walking on a concrete surface, the “concrete” switch will become active and its corresponding sounds will play. If the character then moves from a concrete surface to a grassy surface the “grass” switch will become active and its corresponding sounds will play.

The following illustration demonstrates how the active switch determines which footstep sound is played.



Working with Switches



A switch needs to belong to a switch group before it can be used in Wwise. By grouping switches into a switch group, you can efficiently manage the alternate sounds, music, and motion objects that are available in the game. For example, to manage the different footstep sounds of your character, you can create a switch group called Ground Textures. You would then create a switch in Wwise for each ground surface that appears in your game. Based on what you already know about the different surfaces in the game, you could add switches for gravel, grass, and concrete.



After you have created your switch groups and switches, you can create switch containers and assign objects to the switches. For more information about

working with switches and switch containers, refer to [Defining the Contents and Behavior of Switch Containers](#).

To help you easily identify a switch or a switch groups in the interface, Wwise uses a unique icon to identify them.

Icon	Represents
	Switch group
	Switch

Working with switches can include the following tasks:

- [Creating a Switch Group](#)
- [Creating a Switch](#)
- [Deleting a Switch Group/Switch](#)
- [Mapping Game Parameter Values to Switches](#)

Creating a Switch Group

It is helpful to organize switches into switch groups in a logical way to make it easier to work with them. You can create all the switch groups that you need in the Game Syncs tab of the Project Explorer.

To create a new switch group for your project:

1. In the Project Explorer, switch to the Game Syncs tab.
2. In the Switches section, do one of the following:

Select a virtual folder or work unit and click the **Switch Group** icon in the Project Explorer toolbar.

Right-click a virtual folder or work unit and click **New Child > Switch Group** from the shortcut menu.

A new switch group is added to the list of switch groups.

3. Replace the default name with one that best represents the switch group.



Note

Each switch group name must be unique, and consist only of letters, digits, and underscores. Use either a letter or underscore as the first character.

4. Continue to add switch groups as needed.

Related Topics

- [Deleting a Switch Group/Switch](#)
- [Creating a Switch](#)

Creating a Switch

Each game element that has alternatives, such as ground surfaces, should have a corresponding switch in Wwise. You can create your switches in the Game Syncs tab of the Project Explorer.

To create a new switch:

1. In the Project Explorer, switch to the **Game Syncs** tab.
2. From the Switch Groups list, do one of the following:

Select a switch group and click the **Switch** icon in the Project Explorer toolbar.

Right-click a switch group and select **New Child > Switch** from the shortcut menu.

The new switch is added to the switch group.

3. Replace the default name with one that best represents the switch.



Note

Each switch name in a switch group must be unique, and consist only of letters, digits, and underscores. Use either a letter or an underscore for the first character.

4. Continue creating switches as needed.

Related Topics

- [Creating a Switch Group](#)
- [Deleting a Switch Group/Switch](#)

Deleting a Switch Group/Switch

You may need to delete a switch or a switch group that you no longer need. Keep in mind that deleting a switch group also deletes all the switches in that switch group.



Note

Deleted switches will no longer be available to the presets and objects that were using them.

To delete a switch or switch group:

1. In the Project Explorer, switch to the **Game Syncs** tab.
2. From the Switch Group list, right-click the switch group or switch that you want to delete, and select **Delete Selection**.

The selected switch or switch group is deleted.



Tip

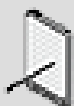
If you delete a switch or switch group by mistake, you can undo the delete by pressing **Ctrl+Z**, or by clicking **Edit > Undo**.

Related Topics

- [Creating a Switch](#)
- [Creating a Switch Group](#)

Mapping Game Parameter Values to Switches

In Wwise, you can also use game parameter values to drive switch changes. After you have created your switch groups and game parameters, you can map game parameter values to switches. For example, if you were using RTPCs to drive switch changes in a car collision, the sounds and motion FX related to the impact would differ depending on the force of the impact. By using the impact force values to trigger switch changes, you can easily ensure that the correct sounds or motion objects play when the collision occurs.



Note

Before you can map game parameter values to switches, you need to create and define your game parameters. For more information on creating and defining game parameters, refer to [Creating a Game Parameter](#).

To map game parameter values to switches:

1. On the Game Syncs tab of the Project Explorer, double-click the switch group that you want to be driven by game parameter values.

The switch group is loaded into the Switch Group Property Editor.

2. Select the **Use Game Parameter** check box.

The Graph view is enabled and the list of switches in the switch group is displayed along the Y axis.

3. From the Game Parameter list, select the game parameter that you want to drive the switch change.

The range of values for the selected game parameter is displayed along the X axis.

4. In the graph view, add a point by double-clicking in the Game Parameter curve along the range and drag that point to the destination switch.

The switch change is mapped to the specified game parameter value.

5. Continue adding points by double-clicking along the curve, and mapping them to switches as needed.

Related Topics

- [Creating a Switch Group](#)
- [Creating a Switch](#)
- [Deleting a Switch Group/Switch](#)
- [Chapter 18, *Working with RTPCs*](#)

Switches Tips and Best Practices

When working with switches, you may want to review the following sections, which provide you with a series of tips and best practices that can help you better manage how you work with switches throughout the audio development process.

Renaming Switches

Before changing the name of a switch, verify how the switches have been integrated into the game. If switches have been integrated using strings containing the name of the switch, changing the name will require additional programming to be able to use the switch.

Deleting Switches/Switch Groups

Before deleting switches or switch groups verify how the switches have been integrated into the game. Switches are integrated using one of two mechanisms:

- By calling an event with a set switch action - deleting a switch or switch group will create problems in Wwise as the called switch will no longer be available.

- By calling the switch group and the switch itself - sound designer who intends to delete the switch or switch group should advise the programmer of the change.

Assigning Objects to More Than One Switch Group

Normally an object can only be assigned to one switch group. If, however, your game requires that an object be assigned to switches in different switch groups, you can assign a second switch group at a higher level in the Master-Mixer, Actor-Mixer, or Interactive Music hierarchies.

Chapter 18. Working with RTPCs

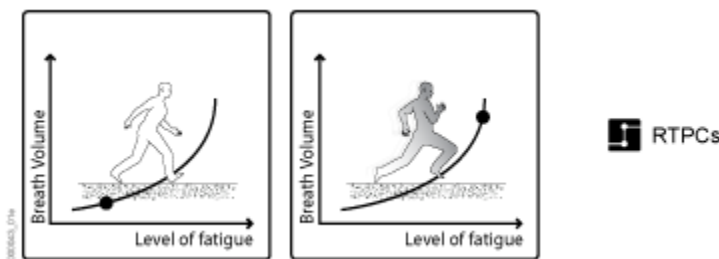
Overview	430
Managing Game Parameters Used in RTPCs	431
Controlling Property Values Using Game Parameters	436
Working with LFOs	443
Working with Envelopes	445
Viewing Game Objects	447
RTPC Tips & Best Practices	447

Overview

To create more dynamic audio and motion in your game, you may want to tie specific object properties to certain parameter values within the game. In Wwise, you can accomplish this using Real-time Parameter Controls. You can create RTPCs using a series of points along a curve. This curve creates a relationship between the game parameter and a property in Wwise. As the game parameter values change in game, Wwise uses the RTPC curve to determine the corresponding property value.

Using RTPCs - Example

Let's say you are creating a first-person shooter game and you want the volume of the main character's breathing to be based on the character's level of fatigue in game. When the character's fatigue level is low, you want the breathing sounds to be very soft and when the fatigue level is high, you want the breathing sounds to be louder. In this case, you can use RTPCs to assign the game parameter (level of fatigue) to the Wwise property (volume). Then using the graph view, you can map the volume levels of the breathing sounds to the main character's level of fatigue in game.



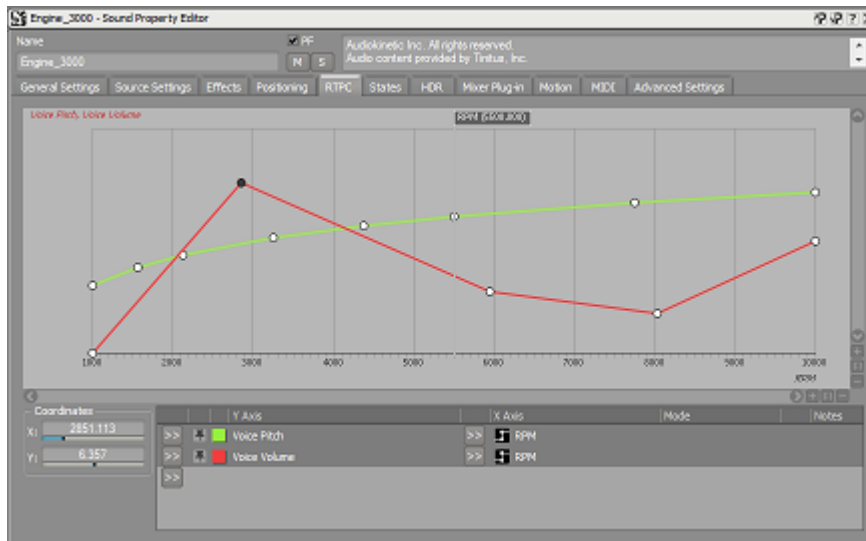
RTPCs can also be used to achieve other effects in your game, such as mapping low pass filter values to water depth, low frequency effect values to the force of an explosion, and so on.

Understanding RTPCs in Wwise

Creating RTPCs in Wwise involves the following:

- [Creating a Game Parameter](#)
- [Assigning Wwise Properties to Game Parameters](#)
- [Mapping Values in the RTPC Graph](#)

The relationship between each property value and game parameter is defined on the RTPC tab of the Property, Effect, or Attenuation Editor.



You can define how you want your object properties to be affected by mapping the game parameter values being sent from the game, which are displayed along the X axis, to the property values in Wwise, which are displayed along the Y axis. An RTPC curve is created by interpolating between the control points that you create.

Each object, bus, attenuation or effect instance can have several curves, where each curve represents a different relationship between an object property and a game parameter. You can display several curves at once for comparison, or just one at a time.

For creating more complex relationships between your object properties and in-game parameters, you can use RTPCs within the blend tracks of a blend container. For more information on using RTPCs within the blend container, refer to [Defining the Contents and Behavior of the Blend Container](#).




Caution

Although RTPCs can be created for all objects, busses, and attenuation and effect instances within your project, it is important to use them selectively as they can consume a significant amount of a platform's memory and CPU.

Managing Game Parameters Used in RTPCs

Before you can map the parameters in your game to the properties in Wwise, you need to create the game parameters first. You can manage the list of game parameters and define their minimum and maximum values on the Game Syncs tab of the Project Explorer.

To help you easily identify a game parameter in the interface, it is represented by the following icon.

Icon	Represents
	Game Parameter

Managing game parameters can involve the following tasks:

- [Creating a Game Parameter](#)
- [Defining the Range of Values for a Game Parameter](#)
- [Deleting a Game Parameter](#)
- [Binding a Game Parameter to a Built-In Parameter](#)

Creating a Game Parameter

If you plan to use a game parameter to drive a sound or motion object's property value, you must create the game parameter first in the **Game Syncs** tab of the **Project Explorer**. After it is created, you can use it to create as many RTPCs as you like.



Caution

When naming game parameters in Wwise, use only letters, digits, and underscores, and make sure that each game parameter name is unique.

You can create all the game parameters that you need in either of the following two places in Wwise:

- **Game Syncs** tab of the **Project Explorer**
- **RTPC** tab of the **Property, Attenuation, or Effect Editor**

To create a new game parameter in the Project Explorer:

1. In the Project Explorer, switch to the **Game Syncs** tab.
2. In the Game Parameters section, do one of the following:

Select a work unit or virtual folder and click the **Game Parameter** icon in the Project Explorer toolbar.

Right-click a work unit or virtual folder and select **New Child > Game Parameter** from the shortcut menu.

A new game parameter is added to the list of game parameters.

3. Replace the default name with one that best represents the game parameter.




Note

Use only letters or underscores for the first character.

4. Continue to add game parameters as needed.

To create a new game parameter from the RTPC tab:

1. On the RTPC tab of the Property, Attenuation, or Effect Editor, select an entry in the RTPC list.

If the entry does not yet have a property selected, select one via the selector button. 

2. Select the **Game Parameters** menu item, then select **New**.

The New Game Parameter dialog box opens.

3. Select the work unit into which you want to create the new game parameter.
4. In the Name field, replace the default name with one that best represents the game parameter.
5. Click **OK** to create the new game parameter.

Related Topics

- [Defining the Range of Values for a Game Parameter](#)
- [Deleting a Game Parameter](#)
- [Binding a Game Parameter to a Built-In Parameter](#)

Defining the Range of Values for a Game Parameter

After you have created a game parameter, you must define its minimum and maximum values. In the case of a racing car, for example, the minimum and maximum speeds would be 0 and 300 km/h.

You can also specify a default game parameter value. This allows you to set a global value to all game objects that do not explicitly specify a particular value. The default game parameter value defined in Wwise will be ignored in the following situations:

- If a game object explicitly specifies a particular value.
- If your game programmer defines a global RTPC value in the SDK.

To define the range of values for a game parameter:

1. In the Project Explorer, switch to the **Game Syncs** tab.

2. In the Game Parameters section, double-click the game parameter whose values you want to define.

The game parameter is loaded into the Property Editor.

3. Define the parameter range by specifying minimum and maximum values.
4. In the Default text box, specify a global value that you want game objects to use if they do not explicitly specify a particular value.

Related Topics

- [Creating a Game Parameter](#)
- [Deleting a Game Parameter](#)
- [Binding a Game Parameter to a Built-In Parameter](#)

Deleting a Game Parameter

You can delete a game parameter that you no longer need in your project. When you delete a game parameter, it will no longer be available to the objects, events, and presets that were using it.



Note

Before deleting a game parameter, be sure to speak to your audio programmer to ensure that it doesn't impact the game code.

To delete a game parameter:

1. In the Project Explorer, switch to the Game Syncs tab.
2. In the Game Parameters section, right-click the game parameter that you want to delete and select **Delete Selection**.

The game parameter is deleted from Wwise.

Related Topics

- [Creating a Game Parameter](#)
- [Defining the Range of Values for a Game Parameter](#)

Binding a Game Parameter to a Built-In Parameter

The sound engine calculates a standard set of values that the sound designer may find useful to hook into to create dynamic audio and motion. These "built-in" parameters can be accessed by using the bind to built-in parameter

mechanism and are updated internally each frame. No additional game programming is necessary to use game parameters that are bound to built in parameters.



Note

When an RTPC is bound to a built-in parameter, the sound engine updates the value for each game object. If the RTPC is used on a global Wwise object that has no game object association, such as a bus or bus effect, the default value will be used.

Available built-in parameters:

- **Distance**

The distance between the game object and the listener. In the case that multiple listeners and/or multiple positions are assigned to the game object, the value is taken to be the shortest distance between all listener and sound position combinations.

- **Azimuth**

The angle, in degrees, between the listener and the game object projected on to the horizontal plane. A value of 0 degrees indicates that the sound is directly in front of the listener, -90 degrees the sound is to the left, 90 degrees to the right and +/- 180 degrees the sound is directly behind the listener.

In the case that the game object has been assigned multiple listeners and/or sound positions, the value taken is the angle between the listener and sound position that are closest together.

- **Elevation**

The vertical angle with respect to the horizon in degrees, between the listener and the game object. A value of 0 degrees indicates that the sound is on the same horizontal plane as the listener; a value of 90 degrees indicates that the sound is directly above and -90 degrees indicates the sound is directly below.

In the case that the game object has been assigned multiple listeners and/or sound positions, the value taken is the angle between the listener and sound position that are closest together.

- **Object-to-Listener Angle**

Object-to-Listener Angle represents the 3D angle between the orientation vector of the game object, and the vector formed by the line between the game object and the listener. A value of 0 degrees indicates that the game

object is directly facing the listener, and a value of 180 degrees indicates that the game object is facing directly away from the listener.

In the case that the game object has been assigned multiple listeners and/or sound positions, the value taken is the angle between the listener and sound position that are closest together.

- **Obstruction**

Obstruction provides access to the value set on the game object via the `SetObjectObstructionAndOcclusion` API.

In the case that the game object has been assigned multiple listeners, the obstruction value taken is the one that has been assigned to the listener with the closest current sound position.

- **Occlusion**

Occlusion provides access to the value set on the game object via the `SetObjectObstructionAndOcclusion` API.

In the case that the game object has been assigned multiple listeners, the occlusion value taken is the one that has been assigned to the listener with the closest current sound position.

Related Topics

- [Creating a Game Parameter](#)
- [Defining the Range of Values for a Game Parameter](#)
- [Deleting a Game Parameter](#)




Controlling Property Values Using Game Parameters

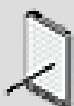
After your game parameters are created, you can begin creating RTPCs for the objects, busses, effect instances, attenuation instances, and switch groups in your project. RTPCs are defined on the RTPC tab of the Property, Effect, and Attenuation Editors or within a blend track of a blend container. Creating an RTPC involves the following steps:

- [Assigning Wwise Properties to Game Parameters](#)
- [Mapping Values in the RTPC Graph](#)

In the graph view, you create the RTPC curves that map property values to the game parameter values. Since you can display many curves in the graph view at the same time, Wwise uses a different color for each curve. The three relative properties (volume, pitch, and low pass filter) are always represented by the same colors, but any other properties within your project are arbitrarily assigned a color. These arbitrary colors may differ from one work session to the next.

The following table shows you which colors have been assigned to the relative properties.

Wwise Property	Color
Volume	(Red) 
Pitch	(Green) 
Low pass filter	(Blue) 



Note

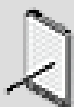
Not all properties are used by all objects and on all platforms; for example, the Pitch property does not affect music objects, the LPF property does not affect motion objects.

Even with the color coding, you may find it useful to hide certain curves to focus on one curve in particular. For more information on hiding curves, refer to [Displaying Curves in the Graph View](#).

Before using RTPCs in your project, it is important to understand how the RTPC values will interact with existing property values. When RTPCs are applied to existing property values, the final property value is determined in one of two ways:

- **Absolute** - The values determined by the RTPC will be used and the object's existing property values will be ignored.
- **Relative** - The values determined by the RTPC will be added to the object's existing property values.

The method used to determine the final property value depends on whether the RTPC property is absolute or relative. If the property is absolute, the original property control is disabled. If relative, the original property control remains available.



Note

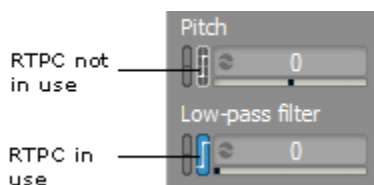
The absolute and relative settings are predefined and cannot be changed.

Assigning Wwise Properties to Game Parameters

After you have created the game parameters and defined their range of values, you can then determine which game parameters will be assigned to which properties.

When you assign a property to a game parameter, the following occurs:

- The X and Y axes are displayed in the Graph view.
- A default RTPC curve is created in the Graph view.
- The RTPC icon beside the property value turns blue.



To assign object properties to game parameters:

1. Load an object, bus, effect or attenuation instance into its Editor.
2. Switch to the RTPC tab.
3. Click the Selector button (>>) and select a property from the list.

A new RTPC curve is created and assigned a unique color. The Wwise property is also assigned to the Y axis in the graph view.

4. From the X Axis list, select the game parameter that you want to assign to the Wwise property.

The game parameter is assigned to the X axis in the graph view.

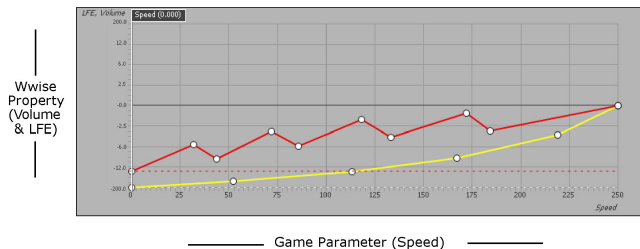
Related Topics

- [Controlling Property Values Using Game Parameters](#)
- [Mapping Values in the RTPC Graph](#)
- [Removing RTPCs From the List](#)

Mapping Values in the RTPC Graph

After the Wwise property is assigned to a game parameter, you can begin to define their relationship in the Graph view. You map property values to game parameter values using control points. For example, if you want the volume of the sound to be at 0 dB when a car is travelling at 250 km/h, you would add a control point at the intersection of 250 km/h and 0 dB. The RTPC curve is created by interpolating between two control points.

The following RTPC graph shows how game parameter values, in this case “Speed”, can be mapped to the property values in Wwise, in this case “Volume”. The Speed values on the X axis are defined in game and the Volume values on the Y axis are controlled by you.



For information on zooming and panning the graph view, adding, moving, and deleting control points, changing the shape of the curve between points, using linear and dB scaling, and other general information about the Graph view, refer to [Chapter 42, Getting to Know the Graph View](#).



Note

You can copy RTPC curves within and between objects in your project. For more information on copying RTPC curves, refer to [Copying RTPC Curves](#).

To map values in the graph:

1. In the Graph view, double-click anywhere on a curve to create a control point.
2. Do one of the following:
 - Drag the control point to the appropriate X and Y coordinates.
 - Type values directly in the X and Y Coordinates text boxes to specify an exact location for the control point.
3. Continue to add points until the curve is representative of the relationship between the property and the game parameter.



Tip

If you continue to hold the mouse button down after creating the control point, you can immediately drag a point to the appropriate location.

Related Topics

- [Adding Control Points](#)
- [Selecting Control Points](#)
- [Moving Control Points](#)
- [Specifying the Shape of the Curve Between Control Points](#)

- [Controlling Property Values Using Game Parameters](#)
- [Assigning Wwise Properties to Game Parameters](#)
- [Copying RTPC Curves](#)
- [Removing RTPCs From the List](#)

Building Smart Pitch Curves

When you tie the pitch of sounds to game parameters using RTPCs, complex curves are key to creating realistic results. Smart pitch curves allow you to create natural-sounding RTPC pitch curves with a minimum of effort.

Smart pitch curves are based on two variables:

- **Native value:** The game parameter value at which the sound is heard at its original pitch.
- **Subdivision level:** The precision of the curve, graded from 1 to 10.

For example, in a racing game, you could apply a smart pitch curve to your engine sounds. If an engine rev sound was recorded at 2000 RPM, you would create an RTPC pitch curve with the native value at 2000 RPM. Wwise then extrapolates the rest of the points required to create a natural-sounding curve.

When you use smart pitch curves to create a pitch RTPC curve, Wwise uses the following equation to create its linear subsegments:

$$\text{Pitch (in cents)} = 1200 * \log_2 \left(\frac{x}{\text{reference } x} \right)$$

The more subsegments your curve has, the more precise it will be. However, these additional subsegments require additional processing at runtime. To save CPU and memory, you should choose the lowest subdivision level that gives you the results you need.



Tip

For more information on ensuring accurate curves, refer to [Valid Ranges for Smart Pitch Curves](#).

To build a smart pitch curve:

1. Create a pitch curve in the RTPC graph view. For more information on how to do this, refer to [Assigning Wwise Properties to Game Parameters](#).

2. Right-click the pitch curve and select **Build Smart Pitch Curve** from the menu.

The Build Smart Pitch Curve dialog box opens.

3. Enter a native value for the smart pitch curve. This is defined as the property value at which the sound was recorded.
4. Enter a subdivision level for the smart pitch curve. The higher the subdivision level of your curve, the more segments it will have and the more time it will take to evaluate at runtime.
5. Click **OK**.

A smart pitch curve is created.



Note

Smart pitch curves can only be created for pitch values between 4800 and -4800 cents. Beyond that, the curve will “clamp” at one of the extreme values.

Related Topics

- [Adding Control Points](#)
- [Selecting Control Points](#)
- [Moving Control Points](#)
- [Specifying the Shape of the Curve Between Control Points](#)
- [Controlling Property Values Using Game Parameters](#)
- [Assigning Wwise Properties to Game Parameters](#)
- [Mapping Values in the RTPC Graph](#)
- [Copying RTPC Curves](#)
- [Removing RTPCs From the List](#)

Copying RTPC Curves

There may be situations in your project where you want the same RTPC curves to be used across properties and even across objects. Instead of creating each curve separately, you can define the shape of a curve once and then copy and paste it to another property within the same object or even across objects within your project.

RTPC curves can be copied in the following areas of Wwise:

- RTPC tab of objects and busses within the Master-Mixer, Actor-Mixer, and Interactive Music hierarchies.
- RTPC tab of effects.

- RTPC tab of source plug-ins.
- Blend Track Editor.

To copy RTPC curves within and between objects:

1. From the curve list, select the curve you want to copy.



2. Do one of the following:

Right-click the selected curve and select **Copy** from the menu.

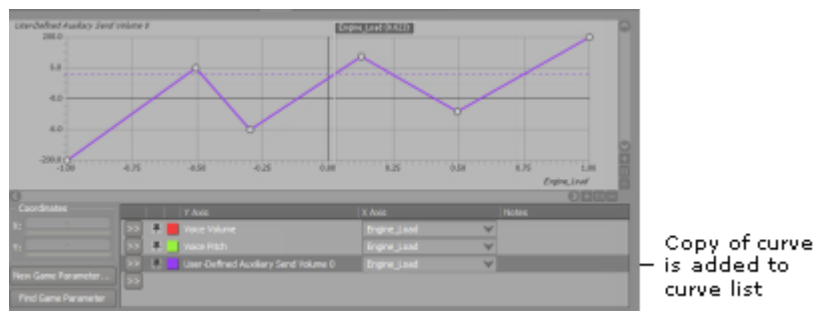
Press **Ctrl+C**.


3. Do one of the following:

Right-click and select **Paste** from the menu.

Press **Ctrl+V**.

An exact copy of the RTPC curve is added to the list.



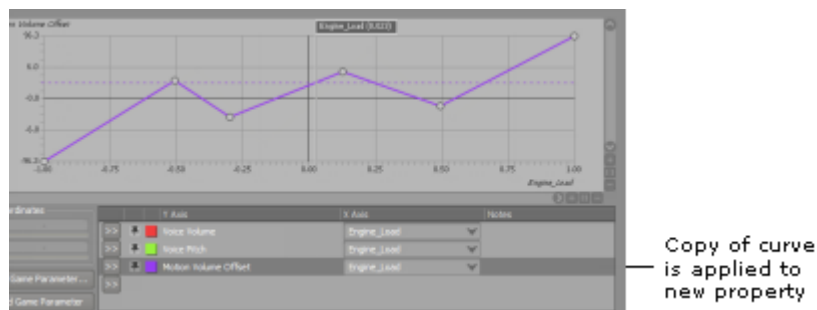


Tip

To paste one or more curves within another object in your project, simply load the new object into the Property Editor, switch to the RTPC tab, and then proceed with Step 3.

4. Click the Selector button (>>) and select any property from the list.

The curve shape is now applied the selected property.



Related Topics

- [Adding Control Points](#)
- [Selecting Control Points](#)
- [Moving Control Points](#)
- [Specifying the Shape of the Curve Between Control Points](#)
- [Controlling Property Values Using Game Parameters](#)
- [Assigning Wwise Properties to Game Parameters](#)
- [Mapping Values in the RTPC Graph](#)
- [Removing RTPCs From the List](#)

Removing RTPCs From the List

When you no longer want a game parameter to drive the values of a particular property in Wwise, you can remove it from the RTPC list.

To delete an RTPC from the list:

1. From the RTPC list, select the RTPC that you want to delete.
2. Press the **Delete** key.

The RTPC is removed from the list.

Related Topics

- [Controlling Property Values Using Game Parameters](#)
- [Assigning Wwise Properties to Game Parameters](#)
- [Mapping Values in the RTPC Graph](#)

Working with LFOs

LFOs (Low Frequency Oscillator) are used to create modulation of property values over time. The properties of the LFO are:

Property	Description
Depth	The amplitude variation of the oscillator (in percentage)
Frequency	The number of cycles per second (in Hz)
Waveform	The shape of the modulator
Smoothing	Low-pass filter over the waveform to smooth hard edges (in percentage)
PWM (Pulse Width Modulation)	The width of the pulse wave, only applies to the Square waveform (in percentage)
Attack	The time it takes for the oscillator to reach full amplitude (in seconds)
Initial Phase	The initial phase of the oscillator waveform (in degrees)
Scope	Define how LFO instances are created: <ul style="list-style-type: none"> • Voice: An instance of LFO is created for every sound/object playback. • Note/Event: An instance of LFO is created for every playing instance, or note when used in MIDI context. • Game Object: An instance of LFO is created for each game object. • Global: A single LFO is created for the whole project.

In Wwise, some properties are additive (Voice Volume, Voice Pitch, etc...), and some are exclusive. When adding an LFO on the additive properties, the LFO modulation is added to the current value of the property. When adding an LFO on the exclusive properties, the LFO modulation replace the current value of the property.

To modulate the Voice Volume with a LFO:

1. In the **Project Explorer**, select an object to add LFO.
2. In the **Property Editor**, go to the RTPC tab.
3. In the RTPC list, click the [>>] button.
4. From the selector menu, select **Voice Volume**.
5. Click the X-axis selector button.
6. From the selector menu, select LFO > Default (Custom).
7. Click the [...] button to edit the LFO properties.
8. Edit the curve to set the range of modulation.

LFO objects can be created as Custom or ShareSet. Custom objects are stored in-place, directly inside the object that has it. ShareSets are stored in a separate work-unit and can be re-used across objects.



Caution

A modulator's processing time depends on its RTPC usage. For most properties, a modulator is evaluated once per audio frame. However, for the property **voice volume**, the associated

modulator is evaluated at every frame. Use modulators selectively as they can consume a significant amount of a platform's memory and CPU.

Working with Envelopes

Envelopes are used to control property values with a predefined shape. An envelope has an ADSR shape:

Property	Description
Attack Time	Defines the time taken for initial run-up of level from nil to peak, beginning when the key is first pressed (in seconds).
Attack Curve	Adjusts the Attack Curve from its linear default slope (50%), to either an exponential-style of envelope (0%) where the rate of change starts slow and then increases or logarithmic (100%) where the rate of change starts fast, then decreases.
Decay Time	Defines the time taken for the subsequent run down from the attack level to the designated sustain level. (in seconds)
Sustain Level	Defines the level during the main sequence of the sound's duration, until the key is released. (in percentage of the range)
Release Time	Defines the time taken for the level to decay from the sustain level to zero after the key is released. (in seconds)
Scope	Define how envelope instances are created: <ul style="list-style-type: none"> • Voice: An envelope instance is created for every sound/object playback. • Note/Event: An envelope instance is created for every playing instance, or note when used in MIDI context.
Trigger On	The actions/MIDI events that may trigger the envelope (i.e. enter the attack phase). <ul style="list-style-type: none"> • Play: either a play action or a MIDI note event. • Note-Off: only a MIDI note-off event
Auto Release	Determines if the envelope requires an action/MIDI event to exit the sustain phase and enter the release phase. If set, the envelope exits the sustain phase after Sustain Time . If not set, the envelope requires an event to exit the sustain phase.
Sustain Time	Defines the time which the envelope will remain in the sustain phase before entering the release phase (in seconds). This value is valid only if Auto Release is set.
Stop playback after release	If set, the playback of the associated sound is terminated once the release phase is complete.

Envelopes can either be used in the context of MIDI or in the context of a normal playback.

When used in the context of MIDI, the envelope is configured for a sound to be played either on note-on or note-off. In the case of a sound to be played on note-on:

- The envelope is configured to trigger on note-on (**Trigger On** parameter).

- The envelope sustains until the first occurrence of:
 - reception of a **Release Envelope** event,
 - reception of a MIDI note-off event,
 - max duration of sustain phase (**Auto Release** is set).

In the case of a sound to be played on note-off:

- The envelope is configured to trigger on note-off (**Trigger On** parameter).
- The envelope sustains until the first occurrence of:
 - reception of a **Release Envelope** event,
 - max duration of sustain phase (**Auto Release** is set).

When used in the general context of a play sound:

- The envelope is configured to trigger on a play action (**Trigger On** parameter).
- The envelope sustains until the first occurrence of:
 - reception of a **Release Envelope** event,
 - max duration of sustain phase (**Auto Release** is set).



Caution

A modulator's processing time depends on its RTPC usage. For most properties, a modulator is evaluated once per audio frame. However, for the property **voice volume**, the associated modulator is evaluated at every frame. Use modulators selectively as they can consume a significant amount of a platform's memory and CPU.

Using Envelopes with MIDI Objects

Envelopes can be used to control property values and also to control the life of sounds. Envelopes have the option of stopping the owner when the release is completed.

To add a Voice Volume envelope to an Actor-Mixer Hierarchy object (instrument):

1. In the **Project Explorer**, select an object in Actor-Mixer Hierarchy
2. In the **Property Editor**, go to the **MIDI** tab
3. Set the **Note-Off** action to **No Action**
4. In the **Property Editor**, go to the **RTPC** tab
5. Click the [**>>**] button in the RTPC list
6. From the selector menu, select **Voice Volume** in menu

7. Click the [\gg] button for the X axis
8. From the selector menu, select **Envelope > Default (Custom)**
9. Click the [...] button to edit the envelope
10. Edit the curve to set the range of modulation.

Viewing Game Objects

Game objects interact with RTPCs in sometimes complex ways. To keep track of these interactions, you can set the RTPC graph to display game objects of interest.

To view game objects within the RTPC graph:

1. Open the **Game Object Explorer**, which is easily accessible via the **Game Object Explorer...** button on the left side of the bottom panel of the RTPC tab view.

All the active game objects, which are created by the audio programmer, are displayed.

2. Add one or more game object watches to the **Game Object Explorer** by following the steps outlined in [Defining Game Object Watches](#).
3. Returning to the bottom panel of the RTPC tab view, select one or more of the listed RTPCs with their associated in-game parameters.

Flags appear in the graph for each of the game parameters.

4. Click **Show Game Objects** in the bottom panel and **Start to Capture** in the Wwise toolbar.

All your watched Game Objects will appear in the RTPC graph.



Note

Without clicking both of these buttons, no game object will appear in the RTPC graph.

Related Topics

- [Monitoring Game Objects and Listeners with Watches](#)
- [Understanding the List of Game Objects](#)

RTPC Tips & Best Practices

Before using RTPCs, you may want to review the following sections, which provide you with a series of tips and best practices that can help you get the most of your sound and motion in game.

RTPC Naming

- Before changing the name of a game parameter, be sure to check how your programmer integrated it into the game engine. If it was integrated using the game parameter's name, you should try to avoid changing it in Wwise as it will require additional work for your programmer.

Performance

- Although RTPCs can be created for all objects, busses, effect and attenuation instances, switch groups, and blend tracks within your project, it is important to use them selectively as they can consume a significant amount of the platform's memory and CPU.

Valid Ranges for Smart Pitch Curves

- Smart pitch curves tend to give good results around their native value, but will not necessarily sound good around extreme values. For example, an engine sound recorded at 2000 RPM will sound perfect at 2000 RPM, and probably quite good in the range from 500 to 3,500 RPM. Beyond that, however, it might not sound natural. One way to remedy this situation is to use multiple recordings at various native values, then assemble the recordings in a blend container. For more information about blend containers, refer to [Defining the Contents and Behavior of the Blend Container](#).

Creating Doppler Type Effects Using RTPCs

The pitch property used in the Wwise Sound Engine pipeline has been highly optimized for re-sampling, allowing for playback to be sped up or slowed down in real time. The “best-practice” approach for creating Doppler type effects in Wwise is to have the game engine keep track of the position delta between the listener and the sound source, which basically equates to a speed value. This 'speed' game parameter can then be mapped to the pitch property of a sound using an RTPC. As the listener and sound move closer or farther away from each other in game, the sound will be pitched up or down. This is by far the least CPU intensive procedure for creating Doppler effects.

When multiple listeners (including split screen) are used in Wwise, Doppler effects using pitch require special design considerations. In real-life, the sound may result in a different pitch value for each listener based on the velocity and distance between the object and each listener. Since in Wwise, game objects use a single instance for each playing sound, having two unique pitch values for a single sound is not possible. Therefore, a single pitch value needs to be determined programmatically.

Chapter 19. Working with Triggers

Overview	450
Working with Triggers	451

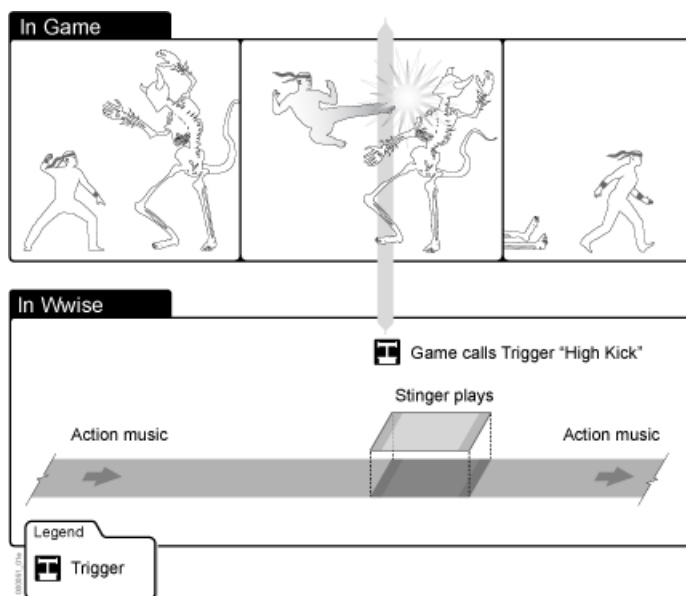
Overview

Like all game syncs, a trigger is a Wwise element that is called by the game and then defines a specific response in Wwise to accommodate what is happening in the game. More specifically, in interactive music a trigger responds to a spontaneous occurrence in the game and launches a stinger. The stinger, which is a brief musical phrase that is superimposed and mixed over the currently playing music, is a musical reaction to the game. For example, when a ninja draws his weapon, you might want to insert a musical sforzando-type effect over the action music already playing to add even more impact to the scene. The game would call the trigger which in turn would launch the stinger and your music clip would play over the ongoing score. For more information about how triggers and stingers work together, refer to [Chapter 28, Using Stingers](#).

Using Triggers - Example

Let's say that you have created a fighting game where your main character is a ninja fighter. At several points in the game your character goes into action mode where he fights his enemies. When your character lands a powerful kick, you want to place a music clip that will intensify the auditory impact of that scene. To build your music for these sequences, you will need to create a trigger, perhaps named "High kick" to be called at these points in the game. In addition, you will define the short music segment that will provide a quick blast of brass to add some "kick".


The following illustration demonstrates the trigger mechanism that plays a stinger at a key point in the game.



Working with Triggers

At key points in the game, a trigger will call a stinger which in turn will play a short music segment over the ongoing music. Stingers are created by mapping individual triggers to music segments and defining playback settings. You can manage the list of triggers in the Game Syncs tab of the Project Explorer.

To help you easily identify a Trigger in the interface, Wwise uses a unique icon to represent it.

Icon	Represents
	Trigger

Managing a trigger involves the following tasks:

- [Creating a Trigger](#)
- [Deleting a Trigger](#)

Creating a Trigger

You can create all the triggers that you need for your project in the Game Syncs tab of the Project Explorer.

To create a new trigger in the Project Explorer:

1. In the Project Explorer, switch to the **Game Syncs** tab.
2. In the Triggers section, do one of the following:

Select a work unit or virtual folder and click the **Trigger** icon in the Project Explorer toolbar.

Right-click a work unit or virtual folder and select **New Child > Trigger** from the shortcut menu.

A new trigger is added to the list of triggers.

3. Replace the default name with one that best represents the trigger.



Note

Each trigger name must be unique, and consist of only letters, digits, and underscores. Use either a letter or underscore as the first character.

4. Continue to add triggers as needed.

Related Topics

- [Deleting a Trigger](#)
- [Chapter 28, *Using Stingers*](#)

Deleting a Trigger

You may need to delete a trigger that you no longer require. Keep in mind that if you delete a trigger that is assigned to a stinger, the association with the music segment for a stinger will be removed.

To delete a trigger:

1. In the Project Explorer, switch to the Game Syncs tab.
2. In the Triggers section, right-click the trigger that you want to delete and select **Delete Selection**.

The selected trigger is deleted.

Related Topics

- [Creating a Trigger](#)
- [Chapter 28, *Using Stingers*](#)

Chapter 20. Working with States and State Groups for Dynamic Dialogue

Overview	454
Working with State Groups	455

Overview

Many games today, especially sports games, have an audio component that is dynamic, or driven by the action that is taking place in game. To build authentic dialogue using traditional methods, you would need to create thousands of assets and build complex switch container hierarchies for each possible scenario. With memory consumption at a premium, you need an efficient way to manage the assets in your project.

To overcome these challenges, Wwise introduces a different approach to manage dynamic dialogue. Every possible condition or outcome in the game is pre-defined in Wwise using states and state groups. State groups can be used to represent the different categories that exist in your game. For example, in a football game, the list of state groups could include Teams, Players, and Actions. Each state group or category also needs a set of corresponding state values. In our football example, the Teams state group could include state values, such as Dallas, Pittsburgh, New England, and so on.

The state groups and states are arranged into dialogue events where every possible game condition is re-created. These conditions called paths are then assigned to a particular voice object. As the game is being played, the current states are matched with those created in the dialogue events in Wwise to determine what piece of dialogue to play.

Using States with Dynamic Dialogue - Example

Let's say you are creating a golf game that will have a play-by-play commentary. You will need to create state groups for each of the different categories in your game. Each state group will then need all the different states that correspond to that category. For our golf game, we will need a variety of state groups including Players, Clubs, Courses, Shots, Locations, Reactions, and so on.

The following illustration shows you how you could divide up some of the different categories in a golf game into state groups and corresponding states.



Arguments	Players	Clubs	Shots	Locations
Argument Values	Woods	Driver	Slice	Fairway
	Ames	Iron	Hook	Tee
	Cabrera	Wedge	Shank	Rough
	Garcia	Putter	Chip	Green

After the state groups and states are defined, you can start adding them to the dialogue events that are required for your game. For more information on creating dialogue events, refer to [Creating Dialogue Events](#).

Working with State Groups

State Groups represent the different categories that can exist in your game, for example Teams and Players in a sports game, or Friends, Enemies, and Weapons in an action adventure game. You can manage the list of state groups in the Game Syncs tab of the Project Explorer.

To help you easily identify state groups and states in the interface, Wwise uses a unique icon to represent them.

Icon	Represents
	State Group
	State Group value

Managing a state group involves the following tasks:

- [Creating a State Group](#)
- [Creating a State](#)
- [Deleting a State Group or State](#)

Creating a State Group

You can create all the state groups that you need for your project on the Game Syncs tab of the Project Explorer.

To create a new state group in the Project Explorer:

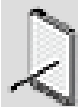
1. In the Project Explorer, switch to the **Game Syncs** tab.
2. In the State Groups section, do one of the following:

Select a virtual folder or work unit and click the **State Group** icon in the Project Explorer toolbar.

Right-click a virtual folder or work unit and select **New Child > State Group** from the shortcut menu.

A new state group is added to the list of state groups.

3. Replace the default name with one that best represents the state group.



Note

Each state group name must be unique, and consist of only letters, digits, and underscores.

4. Continue to add state groups as needed.



Tip

You can double-click a state group to load it into the Property Editor where you can add information about the state group in the Notes field.

Related Topics

- [Creating a State](#)
- [Deleting a State Group or State](#)

Creating a State

Each state group that you create can have many different values. These values represent the different options within a state group category. For example, the states for the **Player Name** state group would be the names of each player in the game. You can create all the states that you need for your project on the **Game Syncs** tab of the **Project Explorer**.

To create a new state in the Project Explorer:

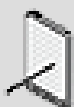
1. In the Project Explorer, switch to the Game Syncs tab.
2. In the State Groups section, do one of the following:

Select a state group and click the **State** icon in the Project Explorer toolbar.

Right-click a state group and select **New Child > State** from the shortcut menu.

A new state is added to the list of state groups.

3. Replace the default name with one that best represents the state group value.



Note

Each state name within a state group must be unique.

4. Continue to add states as needed.



Tip

You can double-click a state to load it into the **Property Editor** where you can add information about the state in the **Notes** field.

Related Topics

- [Creating a State Group](#)
- [Deleting a State Group or State](#)

Deleting a State Group or State

You may need to delete a state group or state that you no longer need. When you delete a state group, all states within that state group will also be deleted. Keep in mind that if you delete a state group, it will be removed from all dialogue events that use it.

To delete a State Group:

1. In the **Project Explorer**, switch to the **Game Syncs** tab.
2. In the **State Groups** section, right-click the state group or state that you want to delete and select **Delete Selection**.

The selected state group or state is deleted.



Tip

You can also select the state group or state and press the Delete key.

Related Topics

- [Creating a State Group](#)
- [Creating a State](#)

audiokinetic

Part V. Creating Interactive Music



21. Understanding Interactive Music	461
Overview	462
Understanding Interactive Music	462
Interactive Music Tips & Best Practices	464
22. Building the Interactive Music Hierarchy	467
Overview	468
What is a Music Segment?	468
Types of Containers	470
About Properties in the Interactive Music Hierarchy	471
Adding Objects to the Interactive Music Hierarchy	472
Adding Parent Objects	474
Managing Music Objects in the Interactive Music Hierarchy	476
Building Interactive Music Hierarchies Tips & Best Practices	477
23. Defining Music Object Playback Behaviors	478
Overview	479
Defining the Time Settings for Music Objects	479
Defining the Playback Behavior of Music Playlist Containers	480
Defining the Contents and Behaviors of Music Switch Containers	484
Interactive Music Playback Tips & Best Practices	488
24. Working with Music Tracks and Segments	490
Overview	491
Adding Music Tracks to Segments	493
Defining the Playback Behavior for Music Tracks	494
Adding Sub-tracks to Tracks	497
Associating Sub-tracks to Switches/States	497
Populating Tracks	498
Removing Tracks and Sub-tracks from Segments	499
Snapping to Increments in the Music Editor	499
Working with Clips	500
Auditioning Segments	502
Working with Cues	504
25. Working with MIDI	508
Creating MIDI Content	509
Importing MIDI files	509
Understanding MIDI content and MIDI target	509
Mixing MIDI and Audio contents	511
Understanding MIDI Tempo	511
Changing the Playback Speed of MIDI	512
26. Creating MIDI Instruments	513
Designing a Synth One Instrument	514
Designing a Simple Sampled MIDI Instrument	514
Understanding MIDI Note Tracking	515
Understanding the MIDI Filters	515
Understanding the MIDI Events	516
Adding Fade-in and Fade-out on MIDI events	517

Using MIDI Data to Control Object Property Values	517
Using the MIDI Keymap Editor	518
Testing an Instrument with a MIDI Keyboard	518
Routing MIDI from a DAW to Wwise	519
27. Working with Transitions	521
Overview	522
Understanding Transitions	523
Adding Transitions	524
Copying and Pasting Transitions	525
Removing Transitions	526
Setting Source and Destination Properties	526
Using Transition Segments	530
Interactive Music Transitions Tips & Best Practices	532
28. Using Stingers	534
Overview	535
Adding Stingers	536
Defining Playback Settings for Stingers	538
Removing Stingers	539
Auditioning Stingers	540

Chapter 21. Understanding Interactive Music

Overview	462
Understanding Interactive Music	462
Interactive Music Tips & Best Practices	464

Overview

Integrating a musical score into a video game presents many unique challenges. A video game score should not only be interactive, it should be straightforward to implement and make the best use of available resources. Wwise's interactive music features present you with tools for dealing with all of these challenges.

Wwise can add interactivity to your game score by allowing you to tie the music being played to the actions taking place in game. Its graphical interface makes it easier for you to arrange and edit pieces of music, as well as determine how and when they will be played in your game. Wwise's interactive music features can also help you conserve resources in many ways. You can save your project's financial resources by extending limited pieces of music over several hours of gameplay. Interactive music features can also free up human resources by speeding up routine composition tasks that would have previously taken composers or other music professionals many hours to do. When it comes to your final product, these features also help you manage console resources allowing you control over the duration, occurrence, and properties of music as it is played.

Understanding Interactive Music

Successful interactive music enhances the gaming experience by providing a varied and meaningful musical score that adapts itself to gameplay. It can get more up-tempo and more driving during combat, for example, or calmer and lighter in peaceful situations. It can also reinforce special events through stingers, or snippets played on top of existing music.

A successful interactive music project can also make the best use of scarce resources. Generally, the ratio of source music to video game score is very low: making one hour of composed music last for thirty hours of gameplay is not unusual. You can use Wwise to organize playback in a way that addresses this need for efficiency while providing variety. You can play individual pieces of music in sequential or random order, or even play the individual tracks that make up these pieces. In this way, you can create the illusion that your game has been provided with many hours of constantly evolving music, even if your original material could cover only a fraction of that time.

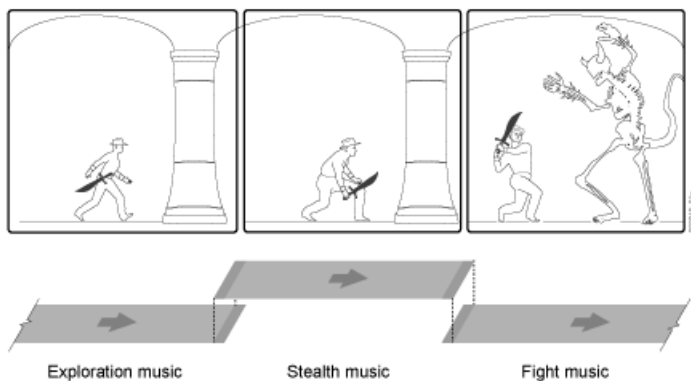
Using Interactive Music - Example

Let's say you are creating an adventure/puzzle game. Your players take on the role of a dashing archaeologist exploring ancient temples in search of artifacts and adventure.

In your game, your players can perform three principal actions: exploring a temple, sneaking past traps, and fighting temple guardians. Using Wwise, you

can define a game state for each action: Exploring, Sneaking, and Fighting. These states define which music Wwise will play at any point in game, ensuring that the soundtrack is adapted to the action.

The following illustration demonstrates the effect you can achieve with interactive music:



You can also use interactive music to make your game music longer and more interesting. After all, if your game offers players beautiful, exotic settings to explore, they might want to spend several hours doing so. Players may become disappointed or even frustrated if the music accompanying their archaeologist's explorations becomes repetitive. You can use interactive music to take a limited amount of source material and extend it.

Overall, interactive music is a multi-faceted tool you can use to elevate your game from a simple pastime to an involving experience. Your players will share the excitement and tension of their archaeologist character, and perhaps even spend a little more time investigating your game's carefully-designed traps and puzzles.

Interactive Music Project Structure

Wwise offers you great flexibility when it comes to putting an interactive music project together. There is an almost infinite number of ways to assemble interactive music objects into a game score. However, following some kind of consistent structure can make your workflow more efficient. Two of the basic structures that can be applied to interactive music projects are as follows:

- A **horizontal project structure** is one in which you re-sequence the game score by shuffling the tracks contained in music segments. This is similar to the track mixing used in music production. It can help you make a varied score out of long, multi-tracked segments.
- A **vertical project structure** is one in which you vary the game score by changing which segments are played at any given time. To do this, you can arrange short discrete segments in the Interactive Music hierarchy, much

like you would arrange objects in the Actor-Mixer hierarchy. In this way, you can make a compelling score from a selection of short music segments while minimizing console requirements.

Typically, you'll use a combination of both these structures to make efficient use of the resources you have available for your project. A good structure lets you show off your music in its best light, and use your console resources to best effect.

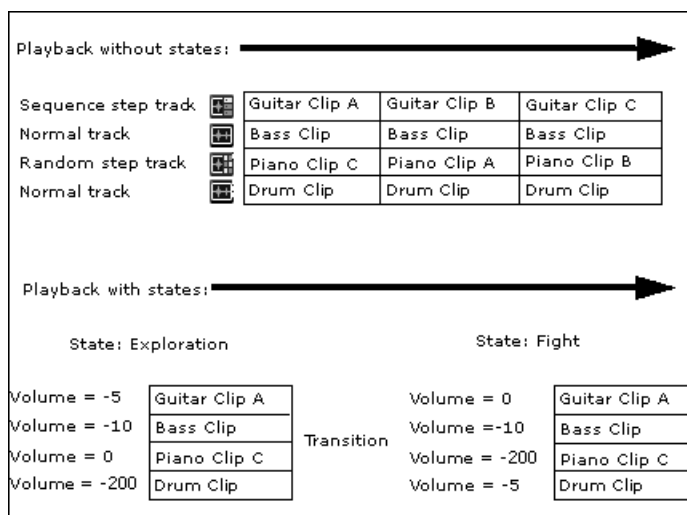
For more ideas on how to structure your project, refer to [Interactive Music Tips & Best Practices](#).

Interactive Music Tips & Best Practices

Interactive music is a complex tool with many options. Adopting a coherent strategy towards interactive music at the beginning of a project can save you time and effort later on. Of course, there are multiple ways to approach any interactive music project, and you can use Wwise in any way you see fit to create the best results for your game. The following are suggestions for how you could implement interactive music.

Implementing a Horizontal Project Structure

A horizontal project structure is one based on the repetition and variation of a small number of segments, each with multiple tracks. A basic horizontal project structure might appear as follows:



In this example, a four-track music segment is used as a basis for interactive music. In the upper part of the illustration, the segment is being played back without any state being specified. The bass and drum clips are set into normal tracks that repeat the same clips each time they repeat. However, because the

guitar track is set to be a sequence step track, the clips in it are played back according to an assigned sequence. The piano track is set to be a random step track, so its clips are played back in random order. In the case of both sequence step and random step tracks, the key to this variety is to create sub-tracks containing alternative clips to the one in the original track.

In the lower part of the previous illustration, the tracks have been modified to respond to state changes. When the game is in the relatively peaceful Exploration state, no drums are heard. Instead, a piano track (made of various sub-tracks) plays. Upon transition to the more exciting Fight state, the volume of the piano track drops to be inaudible, and the drum track volume rises. The guitar volume increases as well. The overall effect is one of music shifting to match the game action.

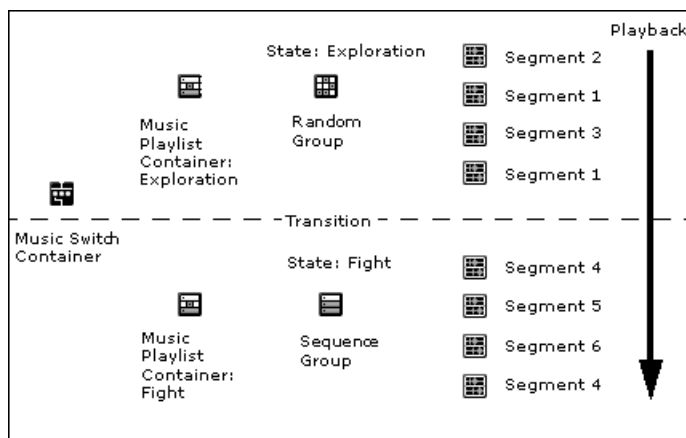
In a horizontal project, you can vary which tracks will play using:

- State changes
- Random step sub-tracks
- Sequence step sub-tracks.

A horizontal approach is particularly suited to projects with a few, complex pieces of music, with relatively infrequent shifts between states.

Implementing a Vertical Project Structure

A vertical project structure is one based on the hierarchical arrangement of several short segments. A basic vertical project structure might appear as follows:



In this example, two music playlist containers hold music segments corresponding to two game states: Exploration and Fight. In the first container, segments are arranged in a random group, so they will play back in random order. In the second container, the segments are arranged in a sequence group, so they will play back according to a predetermined sequence. When

a transition from Exploration state to Fight state occurs, the last segment playing in the Exploration container transitions to the first segment in the Fight container. This creates an effect of complete change from one type of music to another.

In a vertical project, you can vary which segments will play using:

- State changes
- Random containers
- Sequence containers.

A vertical approach is suggested for projects involving many short, relatively simple pieces of music, and is especially good for projects with frequent state changes.

Implementing a Combination Project Structure

The majority of projects created with Wwise use a combination of horizontal and vertical elements. You can decide how much of each type of structure is best suited for your project. For example, you could create a different segment for each of your states (vertical structure), but vary the track playback in each using random sub-tracks (horizontal structure).

Using Multiple Levels of Music Switch Containers

The music switch container supports being bound to multiple state groups or switch groups. In a scenario where the music depends on multiple inputs from the game, it is preferable to avoid creating a hierarchy of cascading music switch container, and instead insert all inputs on the same music switch container. The content of music switch container can be organized using virtual folders, and the association system allow re-using the content from multiple state or switch values across associations.

Be aware that by using multiple levels of switch containers, you introduce a situation where you are incapable of specifying an order of importance for all transitions, because they are defined in multiple distinct sets, both with a catch-all rule, and no relationship of priority exists between them. Consequently, your music transitions will feel random and you will have a hard time tracking all the possibilities.

Chapter 22. Building the Interactive Music Hierarchy

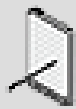
Overview	468
What is a Music Segment?	468
Types of Containers	470
About Properties in the Interactive Music Hierarchy	471
Adding Objects to the Interactive Music Hierarchy	472
Adding Parent Objects	474
Managing Music Objects in the Interactive Music Hierarchy	476
Building Interactive Music Hierarchies Tips & Best Practices	477

Overview

To provide a music score for your game, you will need to manage many musical assets and it is very useful to organize these assets into groups to make this easier. Just as you can group objects together and create parent/child relationships in the **Actor-Mixer Hierarchy**, you can also benefit from organizing all the music assets in your project in the Interactive Music hierarchy. This structure not only organizes your music assets but allows you to define properties and behaviors for the groupings that you create.

You can use a combination of the following music object types to group your audio assets and build a structure for your interactive music project:

- [What is a Music Segment?](#) - one or more music tracks that contain music clips. These clips can be aligned and arranged within the segment.
- [Types of Containers](#) - a group of music segments and/or other music containers defined by their playback behaviors. There are two different types of music containers:
 - Music Switch containers play back music objects according to the switch or state that is called.
 - Music Playlist containers play back in a random or sequence order.



Note

You can also group music objects using virtual folders, but unlike the other music objects, virtual folders don't have any properties or behaviors.

What is a Music Segment?

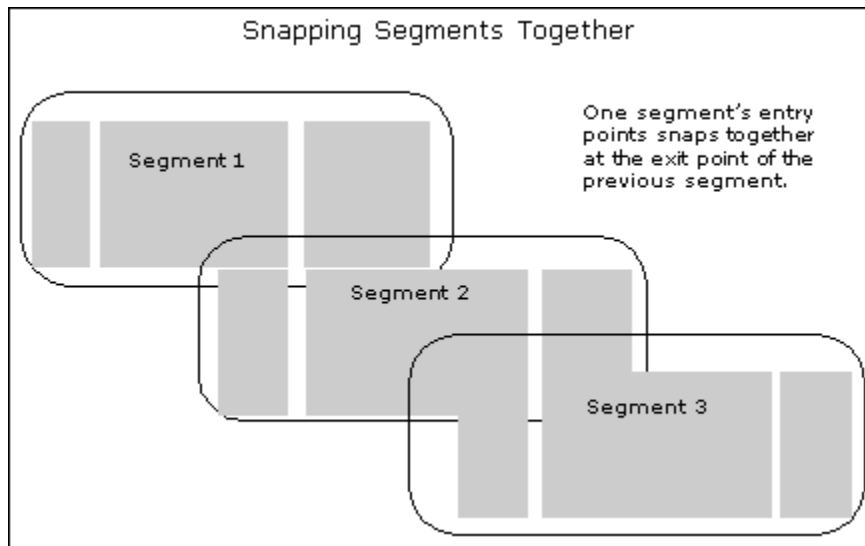
The Wwise music segment is the core music object at the base of the Interactive Music hierarchy. Like the sound object in the Actor-Mixer hierarchy, it contains the audio assets for the music project. However, there are some important differences between a sound object and a music segment:

- The music segment can be aligned with other segments.
- The music segment has another layer between it and its music source - the music track.

Aligning Music Segments

Music segments can be snapped together at defined points to create seamless arrangements of your score. The following illustration demonstrates how the segments can be arranged and snapped together using sync points or cues. Cues are markers appended to segments to indicate key points, such as its entry and

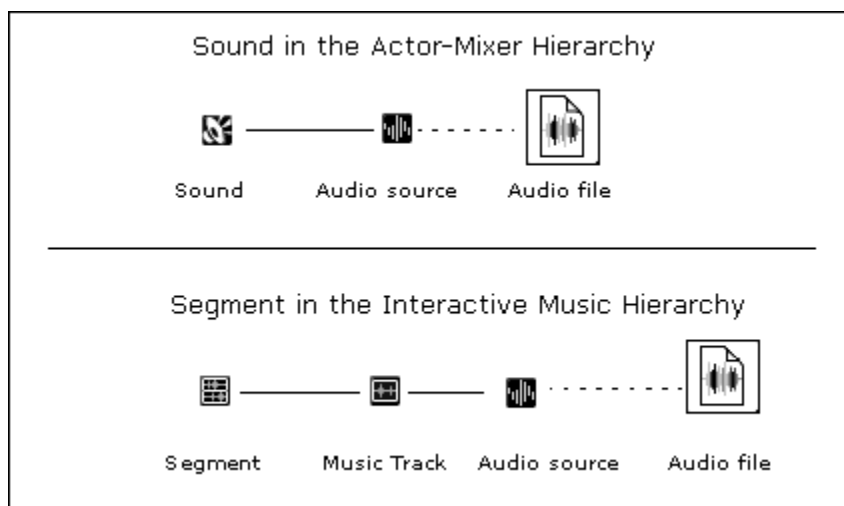
exit points. Using cues and the time settings for tempo and time signature, you can optimally align segments in a playlist as needed.



Music Segments and Music Tracks



Generally, each music segment is composed of at least one track which provides the layer you need to work with your music clips. Each music clip represents an audio source that will contain conversion settings for different game platforms for the corresponding audio file. When you convert your files, your original music files are left intact and new source versions are created per platform. In this way, you can compose interactive music that is optimized for the target platform. For more information about audio file conversion for platforms, refer to [Authoring Across Platforms](#).

The following illustration demonstrates the relationship between an audio file and a sound object, and an audio file and a music segment.



For sound objects, you have the option of adding various audio sources linked to different files and choosing which one will play back. Adding another audio source in a music track, however, adds another clip to the track in the Music Segment Editor.



To help you easily identify a music segment and a track in the interface, Wwise uses a unique icon to identify them.

Icon	Represents
	Music Track
	Music Segment

Types of Containers

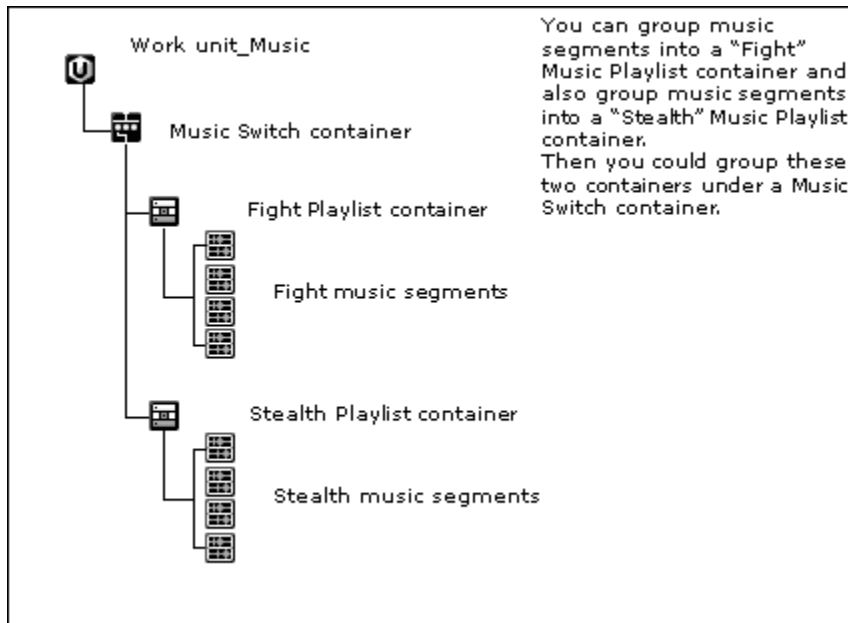
To get the most out of the musical score for your game, you will require containers with different types of playback behaviors. You can define the settings for two types of containers in the Interactive Music hierarchy: music switch and music playlist.

The following table lists the kinds of containers that you can use in the Interactive Music hierarchy.

Icon	Represents
	Music Playlist Container
	Music Switch Container

Organizing Music Assets - Example

Let's say that you are creating the interactive music for a first person action adventure game. You have already established that you will need music for stealth and fighting. So you could group the music objects together in a switch container. That way you can manage the object properties for its child objects. Within your switch container you would add music playlist containers with music objects that are assigned to a switch or state for the container.



You can build your music hierarchy in the early phases of your project based on the game design. You will also need to consider other elements in your project at the same time, such as your work units, routing, game syncs and so on. Looking at your project as a whole can help you group objects together effectively.

For more information about how to build your routing structures, work units, and create game syncs, refer to the following sections:

- [Chapter 5, *Managing Workgroups*](#)
- [Overview](#)
- [Chapter 16, *Working with States*](#)
- [Chapter 19, *Working with Triggers*](#)
- [Chapter 17, *Working with Switches*](#)
- [Chapter 18, *Working with RTPCs*](#)

About Properties in the Interactive Music Hierarchy

Both the Actor-Mixer and the Interactive Music hierarchies manage properties in the same way. In Wwise, the properties of a music object are divided into two categories:

- **Relative properties** - Cumulative properties that are defined at each level of the hierarchy, for volume and low pass filter. The sum of all these values determines the final property value. It is important to know that each relative property will be capped when it reaches either the minimum or maximum limit. The limits for each property are described below:
 - Volume: (-200, +200) in dB
 - Low-pass filter: (0, 100) in percent

- High-pass filter: (0, 100) in percent



Note

Pitch is not considered for objects in the Interactive Music hierarchy for any platform.

- **Absolute properties** - Properties that are defined at one level, usually the highest, in the hierarchy, and then passed down to all child music objects below it, for example, output routing and audio to motion conversion. However, an absolute property can be overridden at each level in the hierarchy.



Note

You can generate motion data from a sound object, including music files. For more information on generating motion from an existing sound object, refer to [Generating Motion from Existing Sounds](#).

Like sound objects, music objects also have property indicators that show whether their property value is linked to other platforms, whether the property value is associated with a game parameter using RTPCs, and whether a Randomizer has been applied on the property value.

For more information on linking/unlinking property values, using RTPCs, and randomizing property values, refer to the following sections:

- [Customizing Object Properties per Platform](#)
- [Controlling Property Values Using Game Parameters](#)
- [Enhancing Sound and Motion by Randomizing Property Values](#)

Adding Objects to the Interactive Music Hierarchy

The **Interactive Music Hierarchy** gives you the flexibility to organize the music objects in your project as you see fit to produce a full musical score. Your starting place for creating your hierarchy is the work unit. If you are working alone you can begin creating your hierarchy directly from the Default Work Unit; otherwise, you will start by creating work units and then add your music objects to the work units. For more information about work units and work groups, refer to [Chapter 5, Managing Workgroups](#).

To actually build the music structure, you can do either of the following:

- Set up your project structure and then import your audio files into the structure.

- Import your audio files and organize them into a project structure that you create afterwards.

For more information on importing audio files and how they create new objects in the Interactive Music hierarchy, refer to [Chapter 6, *Managing Media Files in Your Project*](#).

To create a child object using the Project Explorer toolbar:

1. In the Audio tab of the Project Explorer, select the work unit under which you want to create a music object.

A series of icons become active in the Project Explorer toolbar.

2. From the list, click the icon that represents the object that you want to add.

The object is added to the **Interactive Music Hierarchy** under the selected work unit.

3. Replace the default name with one that best represents the object.



Note

The following characters may not be used when naming music objects in Wwise: ‘:<>%*?”/\|.’

You can now start adding other music objects to the hierarchy. Take the time up-front to understand the relationships between objects so that you can organize them accordingly. This will save you a great deal of time later on in the project.

To create a child object in the Interactive Music hierarchy:

1. In the **Interactive Music Hierarchy** section on the Audio tab of the Project Explorer, right-click the work unit in which you want to create your music object.
2. From the shortcut menu, select **New Child**.

The sub-menu is displayed with a list of objects that you can add.

At this level of the hierarchy, you can add any one of the following objects:

Virtual Folder

Music Switch Container

Music Playlist Container

Music Segment

3. From the list, select the object that you want to add.

The object is added to the Interactive Music hierarchy.

4. Replace the default name with one that best represents the music object.



Note

The following characters may not be used when naming objects in Wwise: ‘:<>*?”%/\|.’

You can now start adding other objects. Take the time up-front to understand the relationships between objects so that you can organize them accordingly. This will save you a great deal of time later on in the project.

Related Topics

- [Adding Parent Objects](#)
- [Managing Music Objects in the Interactive Music Hierarchy](#)

Adding Parent Objects

After you have added the first object to your work unit, you can begin adding more objects to the Interactive Music hierarchy, and create parent-child relationships between them. A parent object contains other objects, and after you create one, you can move existing objects under this new parent. The benefit of creating parent-child relationships is that you can change properties and define behaviors for the parent that will affect the child objects below it. For more information about properties in the Interactive Music hierarchy, refer to [About Properties in the Interactive Music Hierarchy](#).

To create parent object using the Project Explorer toolbar:

1. In the **Interactive Music Hierarchy** section on the **Audio** tab of the Project Explorer, select the music object to which you want to add a parent.
2. Press the **Shift** key to display the icons for the music objects that can be added as a parent of the selected object.
3. From the list, click the icon that represents the music object that you want to add.

The object is added to the **Interactive Music Hierarchy** as the parent of the selected object.

4. Replace the default name with one that best represents the music object.



Note

The following characters may not be used when naming objects in Wwise: ‘:<>*?”%/\|.’

To create a parent object:

1. In the **Interactive Music Hierarchy** section on the Audio tab of the Project Explorer, right-click the object with which you want to create a parent relationship.
2. From the shortcut menu, select **New Parent**.

A sub-menu opens with a list of objects that you can add.

Depending on the selected object's level in the hierarchy, you have the option of adding the following as new parent objects:

Virtual Folder

Music Switch Container

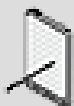
Music Playlist Container

Music Segment

3. From the list, click the object that you want to add.

The new parent object is added to the Interactive Music Hierarchy with a default name, based on its object type.

4. Type the name that best represents the music object.



Note

The following characters may not be used when naming objects in Wwise: ‘:<>*?”%/\|.’

Related Topics

- [Managing Music Objects in the Interactive Music Hierarchy](#)
- [Adding Objects to the Interactive Music Hierarchy](#)

Managing Music Objects in the Interactive Music Hierarchy

In the **Interactive Music Hierarchy** you have access to shortcut menus and the standard Windows shortcuts for renaming, cutting, copying, and pasting objects. Keep in mind the following when you make changes to the hierarchical structure.

Moving Music Objects

Action	Results
Moving a music object	If you change an object's location in the hierarchy, the object will be affected by its new parent's properties and behaviors.
	If you move an object that is associated with an event, the object remains associated with the event.

Copying & Pasting Music Objects

Action	Results
Copying a music object	If you copy and paste a music object in a new location, it will be affected by its new parent's properties and behaviors. Its child objects will be copied as well.
	If you copy a music object that is associated with an event, the new object will not be associated with the event.

Cutting/Deleting Music Objects

Action	Results
Cutting or deleting a music object	If you cut or delete a music object, its child objects will be deleted as well.
	The associated converted audio file is not deleted. Converted audio files that are not associated with an object are called orphan files. To delete these orphan files you need to clear your audio cache. For more information about how to delete orphan audio files, refer to Clearing Your Cache .
	If you delete or cut an object associated with an event, this will result in a missing object in the event.

Related Topics

- [Adding Objects to the Interactive Music Hierarchy](#)
- [Defining the Time Settings for Music Objects](#)
- [Defining the Playback Behavior of Music Playlist Containers](#)
- [Defining the Contents and Behaviors of Music Switch Containers](#)

Building Interactive Music Hierarchies Tips & Best Practices

Interactive music is a complex tool with many options. Adopting a coherent strategy for building your **Interactive Music Hierarchy** at the beginning of a project can save you time and effort later on. Of course, there are multiple ways to approach any interactive music project, and you can use Wwise in any way you see fit to create the best results for your game. The following are suggestions for how you can group objects to get the most from your interactive music.

Grouping Objects in the Interactive Music Hierarchy

Before you start building your music hierarchy you need to think about the best way to organize your objects to save authoring time, but also to manage your project's memory consumption.

To optimize memory usage, consider applying properties at a higher level in the hierarchy so that they can be shared by the entire group. The following properties can be shared:

- Positioning
- RTPCs
- States
- Randomizers

Chapter 23. Defining Music Object Playback Behaviors

Overview	479
Defining the Time Settings for Music Objects	479
Defining the Playback Behavior of Music Playlist Containers	480
Defining the Contents and Behaviors of Music Switch Containers	484
Interactive Music Playback Tips & Best Practices	488

Overview

Different situations within a game will require different kinds of music playback. To accommodate these different scenarios, Wwise allows you to define the playback behaviors of each individual music segment and track. To give you additional flexibility and control over the playback of your music, Wwise also allows you to group objects into two different types of containers: music playlist and music switch containers.

The type of container you choose will determine how the objects within the container will be played back. For example, music playlist containers play back the contents of the container according to a specific playlist. The playlist can contain a series of different groups that can be played back randomly or in a sequential order. Switch containers, on the other hand, play back the contents of the container based on the current switch or state within the game.

You can use a combination of the different containers and playback behaviors to create many different playback scenarios, resulting in less repetition, which can ultimately enhance the music in your game.

Defining the Time Settings for Music Objects

When creating interactive music for games, it can be very useful to time the changes in music with the tempo and meter of the music that is playing. The time settings for your music objects allow you to specify the tempo and meter of the original pieces of music you imported into Wwise. These settings help to label the music objects in Wwise so that you can more easily define sync and transition points for your music.

Along with tempo and time signature, you can also use the grid settings to specify an arbitrary method by which the music segment can be virtually partitioned. By adding another level of granularity to the music segment, you have a great deal of flexibility to determine sync points for music transitions, state changes, and stingers.

You can define the time settings for segments, music playlist containers, and music switch containers.

To define the time settings for music objects:

1. Load a top-level music object into the Property Editor.



Note

If the music object is not a top-level object, you must select the Override parent option before you can define the Time Settings for this object.

2. In the Tempo text box, specify the tempo, in beats per minute, of the original piece of music.
3. In the Time Signature text boxes, specify the number and length of beats to the bar for the original piece of music.
4. From the Frequency list, select an additional method by which your music segment can be partitioned, for example, every 4 bars, beat, whole note, and so on.
5. If you want to create an offset to the frequency value, select a pre-defined option from the list or create a custom offset by selecting Custom and typing a value (in milliseconds) in the corresponding Offset text box.



Note

The Grid settings, both Frequency and Offset, add another level of granularity to the music object, giving you a great deal of flexibility to determine sync points for music transitions, state changes, and stingers.

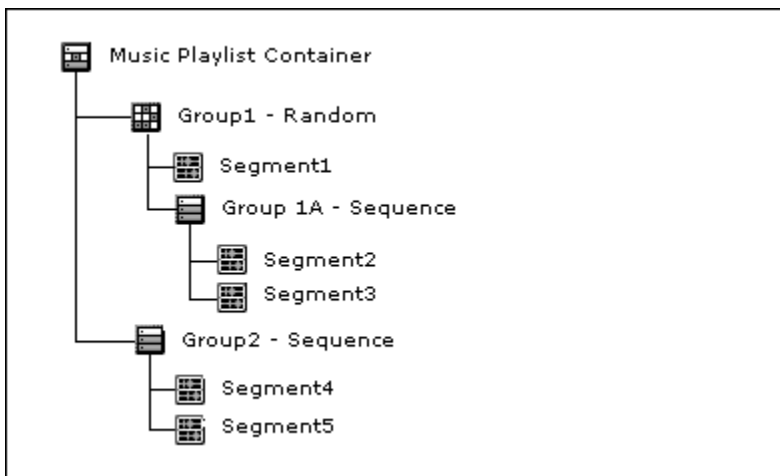
Related Topics

- [Defining the Playback Behavior for Music Tracks](#)
- [Defining the Playback Behavior of Music Playlist Containers](#)
- [Defining the Contents and Behaviors of Music Switch Containers](#)

Defining the Playback Behavior of Music Playlist Containers

To get the most out of the limited amount of music in your project, you can group the music segments into Music Playlist Containers. These music objects are very versatile, allowing you to group different segments in a variety of ways. Each group can have different playback behaviors, using playback modes and types. The playback type determines whether the group will be played back in a random or sequential order. The playback mode determines how many of the segments in the group will be played back each time the group is played.

The following illustration shows an example of how you can use a Music Playlist Container to group some of the music segments in your project.



Using a music playlist container involves the following tasks:

- [Creating Groups within Music Playlist Containers](#)
- [Defining Group Behaviors within Music Playlist Containers](#)
- [Populating a Music Playlist](#)

Creating Groups within Music Playlist Containers

You can create complex music structures by grouping the music segments in your music playlist container. You can add as many groups as you like, and then rearrange them, remove them, cut, copy and paste them, and populate them with music segments. For more information on populating your music playlist container, refer to [Populating a Music Playlist](#).

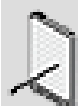
To create a group within a music playlist container:

1. Load a music playlist container into the Property Editor.

A top-level group is automatically created in the Music Playlist Editor.

2. To add a new group in the playlist hierarchy, click **New Group**.

The new group is added under the selected parent object in the hierarchy.



Note

To remove a group from the playlist, right-click the group and select **Delete** or select the group and press the Delete key. The entire contents of the group are removed from the playlist.

3. Continue to add as many groups as you need to your playlist.

At any point, you can define the behaviors of the various groups within your playlist.

Related Topics

- [Defining Group Behaviors within Music Playlist Containers](#)
- [Populating a Music Playlist](#)
- [Defining the Playback Behavior of Music Playlist Containers](#)

Defining Group Behaviors within Music Playlist Containers

For each group within your music playlist container, you can decide between two playback modes and two playback types. The playback type determines whether the group will be played back in a random or sequential order. The playback mode determines how many of the segments in the group will be played back each time the group is played.



Note

The behavior options that will be available for each group will depend on which playback mode and type you have selected for your group.

To define a random group within a music playlist container:

1. Load a music playlist container into the Property Editor.

A top-level group is automatically created in the Music Playlist Editor.

2. From the Group/Segment list select one of the following options to define the playback behavior of the top-level group:

Random Continuous - Plays all music objects in the group one after the other in random order each time the group is played.

Random Step - Plays only one music object in the group each time the group is played.

3. From the Random Type list, select one of the following options:

- **Standard** to keep the pool of objects intact. After a music object is played, it is not removed from the possible list of objects that can be played and can therefore be repeated.
- **Shuffle** to remove music objects from the pool after they have been played. This option avoids repetition of music objects until all objects within the group have been played.

4. In the Avoid Repeat text box, type the number of music objects that must be played before an object within the group can be repeated.

5. In the Loop Count text box, specify the number of times the entire group will be played or the number of steps that will be played.



Note

When you loop a step group, you define the number of objects within the group that will play each time the group is played.

To define a sequence group within a music playlist container:

1. Load a music playlist container into the Property Editor.

A top-level group is automatically created in the Music Playlist Editor.

2. From the Group/Segment list select one of the following options to define the playback behavior of the top-level group:

Sequence Continuous - Plays all music objects in the group in sequential order each time the group is played.

Sequence Step - Plays only one music object in the group each time the group is played. The next time the group is played, the next music object in the group is played.

3. In the Loop Count text box, specify the number of times the entire group will be played or the number of steps that will be played.



Note

When you loop a step group, you define the number of objects within the group that will play each time the group is played. For example, a Sequence Step group with six segments and a loop count of 2 will play segments 1 and 2 the first time the group is played, segments 3 and 4 the second time the group is played, and so on.

Related Topics

- [Creating Groups within Music Playlist Containers](#)
- [Populating a Music Playlist](#)
- [Defining the Playback Behavior of Music Playlist Containers](#)

Populating a Music Playlist

Just like the sequence container in the Actor-Mixer hierarchy, you must specify which segments within the music playlist container will be added to the playlist

and in which order. Each segment must belong to a group that defines the playback behavior of the segments within the group. For more information on defining the playback behavior of groups, refer to [Defining Group Behaviors within Music Playlist Containers](#).

To populate a music playlist container:

1. Load a music playlist container into the Property Editor.
2. From the Contents Editor, drag one or more segments to a group in the music playlist.

The segments are added to the group.

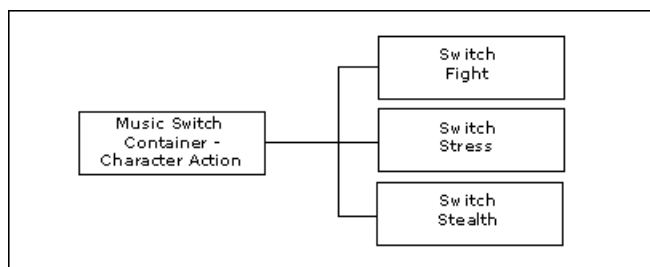
3. If the group is Random Step or Random Continuous, use the Weight text box to assign a weight to each of the music objects within the group. The weight helps to prioritize certain music objects over others.
4. In the Loop Count text box, specify the number of times the segment will be played.
5. Continue to drag segments to the various groups within your music playlist container.

Related Topics

- [Creating Groups within Music Playlist Containers](#)
- [Defining Group Behaviors within Music Playlist Containers](#)
- [Defining the Playback Behavior of Music Playlist Containers](#)

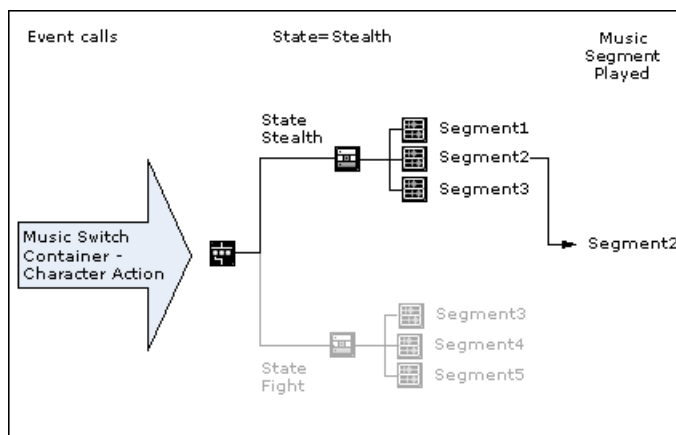
Defining the Contents and Behaviors of Music Switch Containers

Music switch containers allow you to group pieces of music according to the different alternatives that exist for particular elements within a game. Each alternative is represented in the switch container by a switch or state. For example, a music switch container can be created for all the different types of action that can exist for your main character in the game. The container might contain switches for fight sequences, stressful situations, and when the character is in stealth mode.



Within each switch/state are the music objects related to that particular alternative. In this example, all the music segments related to fight sequences in the game would be grouped into the “Fight” switch, all the music segments related to stressful situations would be grouped into the “Stress” switch, and so on. When the game calls the switch container, Wwise verifies which switch or state is currently active to determine which container or music segment to play.

The following illustration demonstrates what happens when an event calls a music switch container called “Character Action”. This container has grouped the music according to the different types of character action that can occur in game. In this example, there are two states: Stealth and Fight. When the event calls the music switch container, the character is in stealth mode (State=Stealth), so the music associated with stealth mode is played. A music playlist container is used to group the different pieces of music within the stealth state so that different variations of the music can be played each time the character is in stealth mode.



Associating the Music Switch Container to one or more Game Syncs

The music switch container can be based on states, switches, or game parameters. To link a music switch container to a game parameter, you need to link a switch groups to a game parameter in the Switch Group Property Editor. For more information on how RTPCs can be associated with switches, refer to [Mapping Game Parameter Values to Switches](#).

The first thing you need to do to set up the music switch container is to assign one or more switch groups and/or state groups to the container. This defines the switches, states, or RTPCs to which the music will react.

Before you can assign a state or switch group to the container or use RTPCs for switches, you must create them first. For information on creating switch groups, state groups, and RTPCs, refer to:

- [Chapter 16, Working with States](#)
- [Chapter 17, Working with Switches](#)
- [Creating a Game Parameter](#)

To associate a game sync to a music switch container:

1. Load a music switch container into the Music Editor.
2. Do one of the following:

Click the [>>] button in the top area of the editor to select a game sync.

Drag and drop a Switch Group or a State Group object from the Project Explorer to the top area of the editor.

Related Topics

- [Defining the Contents and Behaviors of Music Switch Containers](#)
- [Associating Music Objects with States and/or Switches](#)
- [Defining the Time Settings for Music Objects](#)

Defining the Behaviors of Music Objects within Music Switch Containers

If you are re-using different parts of your music, your switches or states may contain several of the same music objects. If this is the case, you will need to decide what the playback behavior of these common music objects will be when a state change occurs. Depending on the situation, you may want the music object to continue playing or to stop at the next sync point and restart from the beginning.

In the following illustration, the Stealth and Stress switches are associated with the same child playlist (S2_Playlist). When the game switches from Stealth to Stress and vice versa, you have the option to continue playing the S2_Playlist or to stop the current music segment at the next sync point and restart the playlist from the beginning.

Switch/State	Child
<input type="checkbox"/> None	
<input type="checkbox"/> Fight	 F_Playlist
<input type="checkbox"/> Stealth	 S2_Playlist
<input type="checkbox"/> Stress	 S2_Playlist

To define the playback behaviors of music objects within a music switch container:

1. Load a music switch container into the Property Editor.
2. In the Play Options group box, do one of the following:

Select the **Continue to play on switch change** option if you want to force the music object that is in both the source and destination switches/states to continue playing during a change in switch/state.

Clear the **Continue to play on switch change** option if you want the music object that is in both the source and destination switches/states to stop playing at the next sync point and to start playing again from the beginning.



Note

You can define the exact sync point and use fades to make the transition seamless.

Related Topics

- [Defining the Contents and Behaviors of Music Switch Containers](#)
- [Associating the Music Switch Container to one or more Game Syncs](#)
- [Associating Music Objects with States and/or Switches](#)
- [Defining the Time Settings for Music Objects](#)

Associating Music Objects with States and/or Switches

After you have assigned at one or more state or switch groups to the music switch container, you can associate different music objects to various combinations of states or switches within the assigned groups.

A particular combination of states or switches, which may also include wildcards, is referred to as a path. Paths are then associated with music objects that are children of the music switch container.

To assign a music objects:

1. Load a music switch container into the Music Editor.

The state groups and switch groups for this container are displayed in the Music Switch Container top area.

2. Select one state for each state group or switch group in the top area.
3. Click the *Add Path* button to create an entry in the entry list.

The entry appears in the entry list, with no object assigned.

4. To assign a music object to the entry, click the [...] button on the entry.

The object browser appears showing the children of the music switch container.

5. Select an object, and click OK.

To assign a music objects using drag and drop:

1. Load a music switch container into the Music Editor.

The state groups and switch groups for this container are displayed in the Music Switch Container top area.

2. Drag and drop one of the Music Switch Container's children over a switch or state on the top area section to create an association with this children.

Related Topics

- [Defining the Contents and Behaviors of Music Switch Containers](#)
- [Associating the Music Switch Container to one or more Game Syncs](#)
- [Defining the Behaviors of Music Objects within Music Switch Containers](#)
- [Defining the Time Settings for Music Objects](#)

Interactive Music Playback Tips & Best Practices

Interactive music is a complex tool with many options. Adopting a coherent strategy towards interactive music at the beginning of a project can save you time and effort later on. Of course, there are multiple ways to approach any interactive music project, and you can use Wwise in any way you see fit to create the best results for your game. The following are suggestions for how you can get the most from your interactive music including troubleshooting information for any playback problems that might arise.

Troubleshooting Playback

If you are having de-synchronization or voice starvation problems when auditioning tracks, and you are connected to the game you may want to set a new look-ahead time for music tracks in the Track Property Editor. If you are experiencing these problems in Wwise for unstreamed tracks, you could adjust the look-ahead time in User Preferences to prevent these problems. This value determines the default look-ahead time for these objects in Wwise. For more information about defining user preferences for look-ahead time, refer to [Setting the Music Track Look-ahead Time](#).

As a general rule, the more tracks that you have playing simultaneously, the more look-ahead time you will need to avoid voice starvation and maintain track synchronization. The exact amount of look-ahead time will depend on many factors, including the compression format and the total bandwidth usage at the time of the request.

Ensuring Sample Accurate Conversion

The sample converter used in Wwise increases the length of a sound by approximately 12 samples per minute for sources in the following frequencies:

- 44,100 Hz
- 22,050 Hz
- 11,025 Hz

Although this is not a problem for sound objects, music objects can stop at the exit marker and don't necessarily play to the end of the file, which can cause inconsistencies. It is a good idea, therefore, to use 12,000, 24,000, 36,000, or 48,000 Hz audio files for your music objects because these don't introduce any errors.






Chapter 24. Working with Music Tracks and Segments

Overview	491
Adding Music Tracks to Segments	493
Defining the Playback Behavior for Music Tracks	494
Adding Sub-tracks to Tracks	497
Associating Sub-tracks to Switches/States	497
Populating Tracks	498
Removing Tracks and Sub-tracks from Segments	499
Snapping to Increments in the Music Editor	499
Working with Clips	500
Auditioning Segments	502
Working with Cues	504

Overview

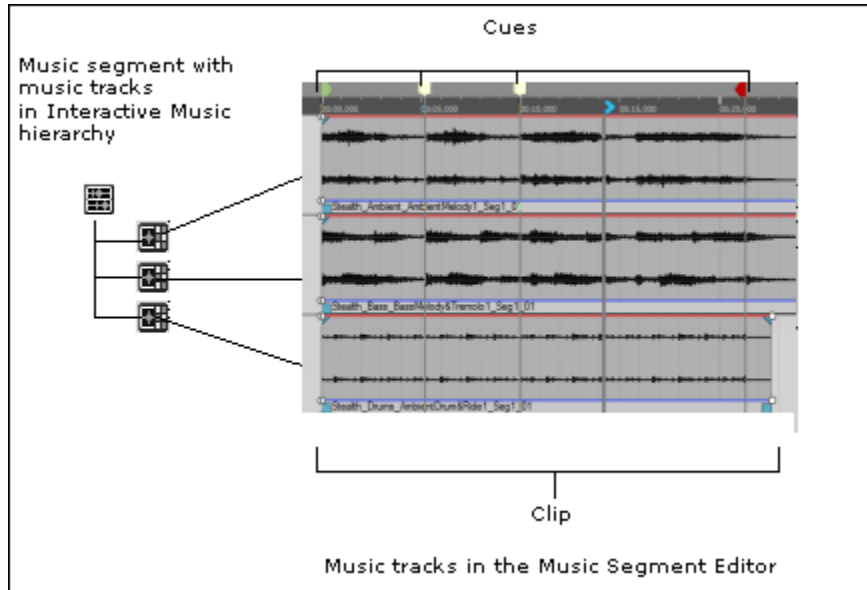
The music segment is the basic unit in the Interactive Music hierarchy. It is like a multi-track sequencer, where tracks and sub-tracks can be arranged to create the layered musical score for your game. Tracks can have an unlimited number of sub-tracks, each of which contains a different clip of music. By assigning different playback behaviors to each track, you can add variety to the music in your game.

To help you easily identify tracks and segments in the interface, Wwise uses a unique icon to identify them.

Icon	Represents
	Music Track - plays back the track each time its parent segment plays.
	Random Music Track - plays back its sub-tracks in random order each time its parent segment is played.
	Sequence Music Track - plays back its sub-tracks in sequential order each time its parent segment is played.
	Switch Music Track - plays back its sub-tracks according to the associated switch group.
	Music Segment - parent object for music tracks.

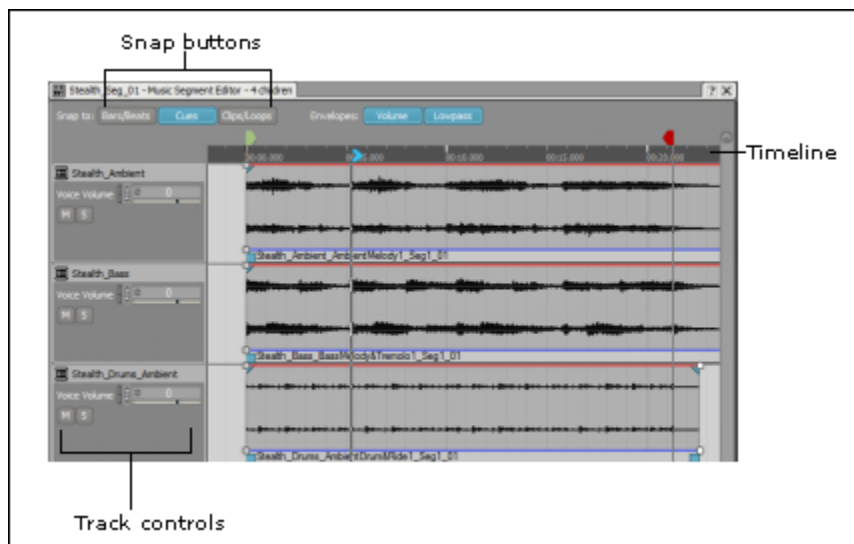
All the tracks in a given segment are displayed in the Music Segment Editor. The music tracks contain music clips that you can view and edit. These clips represent the audio sources and are displayed in waveform along the Music Segment Editor timeline. You can manipulate the clips and add cues to the segment at specific points along the timeline. You can also create custom cues to indicate when property changes or transitions should occur, or when stingers should be played.

Working with Music Tracks and Segments



The Music Segment Editor consists of three areas:

- Snap buttons to align clips and cues to selected elements.
- Track controls to edit the property settings for music tracks, as well as mute and force playback for individual tracks.
- Segment timeline to view clips and cues.



Working with segments includes the following tasks:

- [Adding Music Tracks to Segments](#)
- [Defining the Playback Behavior for Music Tracks](#)
- [Adding Sub-tracks to Tracks](#)
- [Populating Tracks](#)

- [Removing Tracks and Sub-tracks from Segments](#)
- [Auditioning Segments](#)
- [Splitting Clips](#)
- [Working with Cues](#)

Adding Music Tracks to Segments

You can add music tracks to segments from the following locations in Wwise:

- Audio tab of the Project Explorer
- Music Segment Editor



Note

By default a new track will have a Normal behavior. To define a different behavior for a music track, refer to [Defining the Playback Behavior for Music Tracks](#).

To add a new music track from the Audio tab:

1. In the Interactive Music hierarchy, right-click the segment to which you want to add a track and select **New Child > Music Track**.

The new track is added to the segment and is displayed in the Music Segment Editor.

2. Replace the default name with one that best represents the music track.

To add a new music track in the Music Segment Editor:

1. Load a segment into the Music Segment Editor.
2. In the Music Segment Editor, right-click to open the shortcut menu, and select **New Track**.

The New Music Track dialog box opens.

3. Replace the default name with one that best represents the music track and click **OK**.

The new track is added to the Music Segment Editor.

Related Topics

- [Populating Tracks](#)
- [Defining the Track Playback Type](#)

- [Removing Tracks and Sub-tracks from Segments](#)

Defining the Playback Behavior for Music Tracks

Each music track within your project has a certain set of behaviors. The behaviors determine how a music track and its sub-tracks are played back and whether the music on the track is stored in memory or streamed directly from the media or hard drive.

You can define the following behaviors for a music track:

- [Streaming Your Music](#)
- [Defining the Track Playback Type](#)

Streaming Your Music

You can determine which pieces of music will be played from memory and which ones will be streamed from the DVD, CD, or hard drive. When music is streamed from the disk or hard drive, you also have the option to avoid any playback delays and syncing problems by using a combination of the following:

- **Look-ahead time** - Which reserves a specific amount of time so the sound engine can seek the streaming data. This time defines the latency of the track.
- **Prefetch** - Which creates a small audio buffer that covers the latency time required to fetch the rest of the file.

You can specify the amount of look-ahead time and prefetch size so that the requirements of the different media sources, such as hard drive, CD, and DVD are met. A prefetch size that is too small will result in latency and one that is too large will take up too much memory. A look-ahead time that is too small will result in de-synchronizing of music tracks within a segment, whereas a look-ahead time that is too large will result in playback delays. You will need to experiment with the prefetch size and look-ahead times to find the right balance between latency and synchronization of your music clips.

The look-ahead time and the prefetch settings can be used together to reduce the chance of de-synchronization and voice starvation when streaming music in your game. For example, when the sync point for a piece of music is not at the very beginning of the segment or you've chosen not to use the pre-entry portion of the segment, the prefetch data loaded into memory can not be used, so Wwise will use the look-ahead time instead.



Note

Audio playback within the Wwise authoring application is always streamed regardless of whether the streaming options have

been selected. As a result, music objects that are not streamed in game will still be streamed when played back in Wwise. To avoid de-synchronization of music tracks and voice starvation while playing back these music objects in Wwise, you may need to define an internal playback look-ahead time. For more information on setting an internal playback look-ahead time, refer to [Setting the Music Track Look-ahead Time](#).

To stream a music track:

1. Load a music track into the Property Editor.
2. Select the **Stream** option.

The Stream controls become available.

3. In the Look-ahead time text box, type the number of milliseconds that you want to reserve so the sound engine can seek the streaming data. This time defines the latency of the track, so if you specify a value of 100 milliseconds, the music will be heard only 100 milliseconds after pressing Play.



Note

When Wwise seeks the data for a segment that has several tracks with different look-ahead times, Wwise will use the longest look-ahead time of the tracks that have a source to play.

4. Select the **Zero Latency** option to have no delay from the time the music is triggered to when it is actually played.
5. To achieve zero latency, a certain portion of the beginning of the sound must be stored in memory to cover the latency time required to fetch the rest of the audio file from the media drive. In the Prefetch length text box, type the number of milliseconds of the music that you want to store in memory.



Note

Prefetch settings are set per track, so if you have more than one music source on a track, then the beginning portion all sources on the track will be loaded into memory.

Related Topics

- [Defining the Track Playback Type](#)
- [Populating Tracks](#)

Defining the Track Playback Type

To add variety to the music in your game, you can define the playback type for each of the tracks within a segment. You can have a normal track, which simply plays the same track each time the segment is called, or you can create random or sequence tracks. Random step and sequence step tracks can have an unlimited number of sub-tracks, which allow you to play a different piece of music each time the track is played.

The following illustration shows you the differences in playback behavior between the three track types.

Track Type	Segment A	Game calls Segment A (1st time)	Game calls Segment A (2nd time)	Game calls Segment A (3rd time)
		Wwise Plays ↓	Wwise Plays ↓	Wwise Plays ↓
Normal	Track1	Track1	Track1	Track1
Random Step	Track1 Sub-track1 Sub-track2 Sub-track3	Sub-track3	Sub-track1	Sub-track2
Sequence Step	Track1 Sub-track1 Sub-track2 Sub-track3	Sub-track1	Sub-track2	Sub-track3

By using a combination of the three types of tracks, you can give the impression that a new piece of music is played each time the segment is called by the game. For example, you can build a music segment that has four random step tracks, each of which has four sub-tracks. When the music segment is called by the game, you can have 256 different variations of the same music.

To define the track playback type:

1. Load a music track into the Property Editor.
2. From the Track Type group box, select one of the following options:

Normal to play the current track each time the parent segment is played. Sub-tracks can't be added when in Normal mode.

Random Step to play one of the sub-tracks in a random order each time the parent segment is played.

Sequence Step to play one of the sub-tracks in sequential order each time the parent segment is played.

Switch to select which sub-track(s) to play.

Related Topics

- [Defining the Playback Behavior for Music Tracks](#)
- [Streaming Your Music](#)
- [Associating Sub-tracks to Switches/States](#)

Adding Sub-tracks to Tracks

To add variety to your score, you can add layers or sub-tracks to tracks. Since you can define random or sequence behaviors for the track and sub-tracks, this provides you with an almost unlimited number of permutations and combinations for playback. Adding sub-tracks is only available for tracks that have been defined as Random Step, Sequence Step or Switch. For more information about how to work with behaviors and tracks, refer to [Defining the Playback Behavior for Music Tracks](#). After you populate your sub-tracks, you might want to audition some of the sub-tracks individually. For testing purposes in the Wwise authoring application only, you can override the random or sequence behavior for a track by using the Force Usage functionality. For more information about auditioning individual sub-tracks, refer to [Forcing Sub-tracks to Play](#).

To add a sub-track to a track:

1. Right-click the track where you want to add a sub-track.
2. In the shortcut menu, select **Add Sub-Track**.

A new sub-track is added to the track.

Related Topics

- [Populating Tracks](#)
- [Defining the Track Playback Type](#)
- [Adding Music Tracks to Segments](#)
- [Associating Sub-tracks to Switches/States](#)

Associating Sub-tracks to Switches/States

To add variety to your score, you can determine which sub-tracks are played via a switch/state group. To do so, the Track Type must be set to Switch. For more information about how to work with behaviors and tracks, refer to [Defining the Playback Behavior for Music Tracks](#).

To associate a sub-track to a switch/state, a switch/state group must be selected for the track. The track's switch/state group is selected via the controls in the **Switch Type** box of the track's **General Settings** tab.

The behavior of the track upon a switch/state change is set via the Transitions tab. For more information on how to work with transitions, refer to [Chapter 27, Working with Transitions](#).

To associate a sub-track to a switch/state:

1. Right-click the sub-track's header (yellow bar).
2. In the shortcut menu, select **Association**.
3. Select either an existing switch/state, or create a new one.

The sub-track is associated to the switch/state.

The switch/state name is displayed along the top of the sub-track.

Related Topics

- [Populating Tracks](#)
- [Defining the Track Playback Type](#)
- [Adding Music Tracks to Segments](#)

Populating Tracks

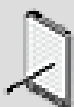
Much of the work that you do in your interactive music project is at the track level where you add your music clips to tracks and then arrange, edit, and mix the clips as needed.

To populate the music tracks, you can use the Audio File Importer to import music files into the tracks just as you would any audio file in Wwise. For more information about importing, refer to [Importing Media Files](#).

Quick Import

The following table lists the shortcuts you can use to import audio files into music tracks.

To	Use this shortcut
Import music files.	Drag the files into a music track in the Project Explorer.



Note

You can also replace your audio files as needed, using the replace functionality. For more information about replacing files, refer to [Replacing Media Files](#).

Related Topics

- [Removing Tracks and Sub-tracks from Segments](#)

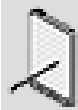
- [Defining the Track Playback Type](#)
- [Adding Music Tracks to Segments](#)

Removing Tracks and Sub-tracks from Segments

You can remove a track or a sub-track from a segment when you do not need it any longer. Deleting a track also deletes all its sub-tracks.

To remove a track or sub-track from a music segment:

1. In the Music Segment Editor, right-click a track or sub-track that you want to remove.



Note

You can also remove a track in the Audio tab of the Project Explorer.

2. In the shortcut menu, select **Delete**.

The selected track and its sub-tracks are deleted from the segment.



Note

Deleting a track or sub-track does not delete the associated converted audio files. Orphan files are only deleted from your project when you clear your audio cache. For more information about deleting audio files, refer to [Clearing Your Cache](#).

Related Topics

- [Populating Tracks](#)
- [Defining the Track Playback Type](#)
- [Adding Music Tracks to Segments](#)

Snapping to Increments in the Music Editor

To help you position your clips, cues, or the play cursor more accurately along the timeline, you can snap them to certain increments that you can define.

To snap along the timeline:

1. To snap the play cursor, a clip, or a cue to any of the following along the timeline, click the corresponding button(s):

Bars/Beats

Cues

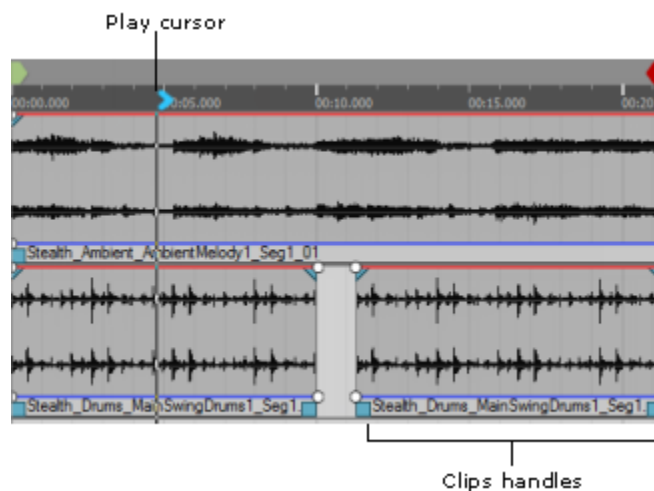
Clips/Loops

Related Topics





- [Working with Cues](#)
- [Working with Clips](#)

Working with Clips

Using the various tools for clips in the Music Segment Editor, you can edit your clips easily along a timeline. You can move clips from one track to another by dragging them up or down and overlap one clip over another. You can make a clip shorter by dragging one of its handles inwards, or extend it by dragging the handle outwards. When you extend a clip, it repeats itself. Each repeat is called a loop.



The following table lists the various clip tools and indicators you will use when working with clips in the Music Segment Editor.

Icon	Name	Description
	Clip handle	Drag this to shorten or extend a clip. When you extend a clip, you create loops.
	Play cursor	When a segment is playing or paused, indicates current playback position within the segment. When a segment is stopped, indicates the point at which playback will begin.
	End cursor	Drag this to shorten or extend all the music tracks in a segment within the Music Segment Editor. Shortening or extending music tracks does not affect the length or playback of clips on these tracks.
	Clip indicator	Indicates the beginning of a clip when a clip is looped.

Looping Clips

You can easily loop your clips using the clip handles in the Music Segment Editor.

To loop a clip in the Music Segment Editor:

1. Click the clip handle and drag the handle to extend the clip.
2. Extend the clip until the clip indicator appears in the timeline.

A clip loop is created. You can continue extending the clip to add as many loops as needed.

Moving Clips

To help layer and arrange your music, you can move a clip in a track. You can line up and overlap the clips in the Music Segment Editor in any way you want to mix your music. Using the Snap functionality you can accurately align the clips along the timeline.

To move a clip:

1. Select a clip in a track and drag it to the new location.



Tip

You can also use standard Windows cut, copy, and paste commands to move clips and entire tracks.

Splitting Clips

You can split the clips in the Music Segment Editor. This can be useful when, for example, you want to isolate a single bass drum hit so that you can repeat it for an entire measure or insert it into another track. You could add this isolated drum bass clip to tracks anywhere and thus rearrange and even re-compose your music.

To split a clip:

1. Drag the play cursor to the location where you want to split the clip.
2. In the track, right-click the clip that you want to split and select **Split on Play Cursor** from the shortcut menu.

The clip is split at the position of play cursor. You can now edit the clips as needed.

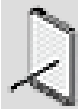
Stacking Clips

When you are overlapping your clips, it can be useful to change how your clips are stacked particularly when you want to view sync points for a clip that has been overlapped. You can do this using the Bring to Front and Send to Back commands on the shortcut menu.

To change the stacking order of clips:

1. In a music track, right-click the overlapping clip, and select **Send to Back**.

The order for the clips is reversed and the clip that was underneath is now on top.



Note

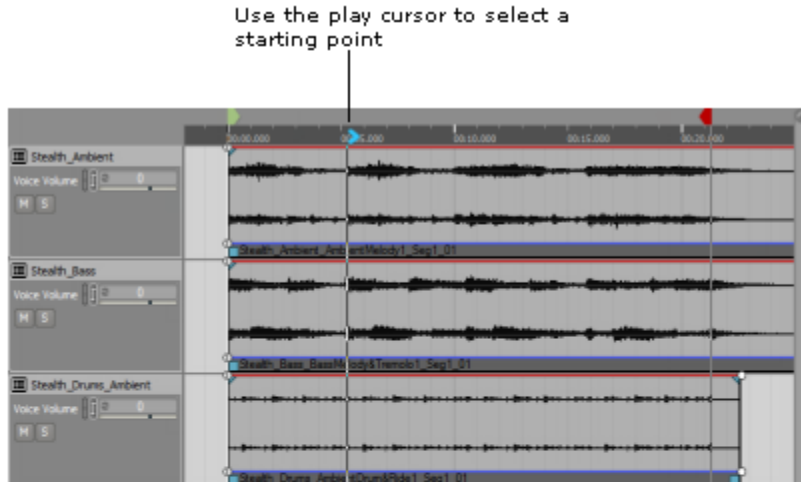
Alternatively, you can also change the stacking order of the clips by right-clicking the visible section of a clip and selecting **Send to Front**.

Related Topics

- [Working with Clips](#)
- [Splitting Clips](#)
- [Moving Clips](#)
- [Looping Clips](#)

Auditioning Segments

After you have populated your tracks in a segment, you can audition your segment tracks at any point in the Transport Control. To play back specific points in a clip you can use the play cursor to indicate where you want the playback to start.



When auditioning your segment you can listen to all the tracks as a whole or you can mute or solo individual tracks to listen to the content of each track as needed. You can also force only one sub-track to play back.

To audition a track in the Transport Control:

1. Load a music segment into the Music Segment Editor.

All the tracks and sub-tracks in the music segment are loaded into the Music Segment Editor and the music segment is loaded into the Transport Control.



2. If you want to audition the entire segment, click **Play** in the Transport Control.
3. To mute or solo a music track, click the **Mute** or **Solo** buttons in the track control area.
4. If you want to play from a location along the track's timeline, drag the play cursor to the location where you want playback to begin.

Related Topics

- [Working with Clips](#)
- [Adding Music Tracks to Segments](#)
- [Adding Sub-tracks to Tracks](#)
- [Forcing Sub-tracks to Play](#)

Forcing Sub-tracks to Play

When a random or sequence track is played back, the sub-tracks play in their random or sequence order. If you are experimenting with clips in a particular sub-track, you might find it useful to be able to audition your changes by overriding the random or sequence behavior for the track and forcing the playback of that sub-track.

To force a sub-track to play:

1. 

In the Music Segment Editor, click the **Force Usage** icon to force that sub-track to play in the Transport Control.

The icon turns blue, and the selected sub-track will play in the Transport Control.

2. To turn off this option, click the **Force Usage** icon again.



Tip

In the Transport Control, click >> to open the Reset menu and select **Reset All Music Tracks Force Usage** to remove this setting.

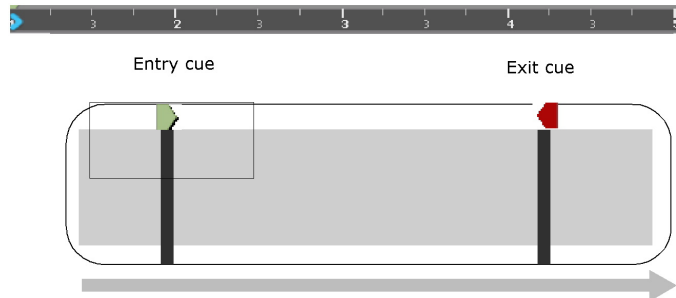
Related Topics

- [Working with Clips](#)
- [Adding Music Tracks to Segments](#)
- [Adding Sub-tracks to Tracks](#)

Working with Cues

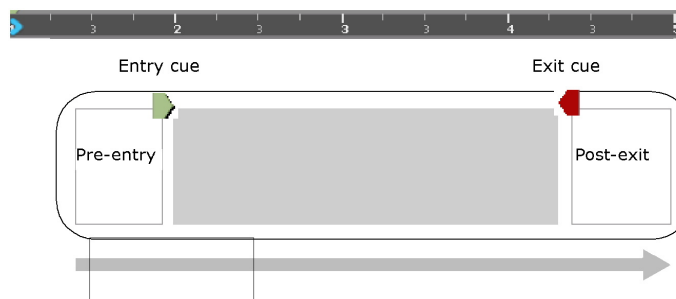
A cue represents a specific point along the timeline. Cues are essentially used as sync points that can help align music segments for the various shifts that

take place in interactive music such as state changes, playing stingers, and transitions. These alignments must take into account the tempo and sample accuracy concerns as well as contextual considerations. All segments have entry cues and exit cues and may also have custom cues.



The entry cue defines where the first beat of the first bar starts and where to snap with the exit cue of a previous segment. The entry and exit cues cannot be removed from the clip, but can be edited.

The segment area before the entry cue is called the pre-entry and the area beyond the exit cue is called the post-exit. These portions of the segment can be played, but this is dependent on the transition settings. For more information about transitions, refer to [Understanding Transitions](#).



In addition to the entry and exit cues, you can add any number of custom cues to help sync your music with what is happening in the game.

Working with cues includes the following tasks:

- [Adding Custom Cues](#)
- [Removing Cues](#)
- [Describing Cues](#)

Adding Custom Cues

You can add a custom cue to any location along the timeline to be used for state changes, transitions or stinger insertions.

To add a cue to a track:

1. In the Music Segment Editor, right-click the timeline ruler where you want to add a custom cue.
2. From the shortcut menu, select **Add Custom Cue**.

A new custom cue is added in the location you selected on the timeline.

3. Drag the cue to the desired location.
4. Continue to add custom cues as needed.

Related Topics

- [Using Entry and Exit Cues](#)
- [Describing Cues](#)
- [Removing Cues](#)

Removing Cues

You can remove custom cues at any time. Entry and Exit cues, however, cannot be deleted.

To remove a custom cue:

1. Along the track timeline, right-click the custom cue that you want to remove.
2. From the shortcut menu, select **Delete**.

The cue is removed from the track.

Related Topics

- [Using Entry and Exit Cues](#)
- [Describing Cues](#)
- [Adding Custom Cues](#)

Describing Cues

You can add a description to represent a cue. This provides a context for the cue. This might be useful when you want to refer to a special cue that you have created for a stinger for example.

To describe a custom cue:

1. Along the track timeline, right-click the custom cue that you want to describe.
2. In the shortcut menu, select **Change Description**.

The Change Description dialog box opens.

3. Replace the default description with one that best represents the cue.
4. Click OK.

Related Topics

- [Using Entry and Exit Cues](#)
- [Removing Cues](#)
- [Adding Custom Cues](#)

Using Entry and Exit Cues

You can move the entry and exit cues to different locations on the timeline and in so doing extend or decrease the pre-entry and post-exit areas in your segments.

To move entry and exit cues for a segment within a selected clip:

1. Select and drag the cue left or right along the segment.

To move entry and exit cues for a segment to the ends of a selected clip:

1. Right-click the clip where you want to define the entry and exit cue locations.
2. From the shortcut menu, select **Move Entry/Exit Cues to Selection**.

The Entry and Exit cues are positioned at the beginning and end of the selected clip.

Related Topics

- [Removing Cues](#)
- [Describing Cues](#)
- [Adding Custom Cues](#)

Chapter 25. Working with MIDI

Creating MIDI Content	509
Importing MIDI files	509
Understanding MIDI content and MIDI target	509
Mixing MIDI and Audio contents	511
Understanding MIDI Tempo	511
Changing the Playback Speed of MIDI	512

Creating MIDI Content

In addition to WAV files, MIDI files may also be used in the Interactive Music hierarchy. Both file types, once imported, are used to create clips, contained in one or many music segment tracks. In the case of WAV clips, each clip is a self-contained source of sound. However, MIDI clips only contain the musical notes to be played; a MIDI instrument must be used to interpret the notes. The instrument interprets all MIDI events generated by one or many MIDI clips targeting it.

Importing MIDI files

MIDI files can be imported to the **Interactive Music Hierarchy** inside Music Clips. MIDI files can't be imported into the **Actor-Mixer Hierarchy**.

A MIDI clip can be created in all the same ways as a WAV clip. For more information on how to create a clip from a file, refer to [Populating Tracks](#).

To import a MIDI file using drag and drop:

1. Drag a MIDI file from Windows Explorer or Mac Finder to any location in the **Interactive Music Hierarchy**.

The **Audio File Importer** opens.

2. Click OK to complete the import operation.



Note

Hold the CTRL key during drag and drop to skip the Audio File Importer dialog.

To import a MIDI file using the Audio File Importer:

1. Select the location to import the .mid in the Project Explorer.
2. Use the keyboard shortcut (**Shift + I**) to open the Audio File Importer.
3. Click the **Add Files...** button
4. Browse for a .mid file, and click OK.
5. Click OK again to complete the import operation.

Understanding MIDI content and MIDI target

The MIDI file reference is stored inside the Music Clips contained in tracks and segments. The MIDI clips only contain the source MIDI information; notes,

pitch bend, CC. They don't define the instrument to play. The instrument to use for playback is defined by the MIDI target reference found on the Music Segment and other music ancestors.

- MIDI playable instruments need to be created and stored in the Actor-Mixer Hierarchy.
- MIDI playable instruments use the standard object structures used for sounds in general.

Example:

- **Actor-Mixer Hierarchy**
 - Instrument Work Unit
 - Drum Kit
 - Piano
- **Interactive Music Hierarchy**
 - Default Work Unit
 - Song Segment
 - Piano Track (with MIDI clips)
 - Drum Track (with MIDI clips)
 - Guitar (with audio clips)

In the previous example:

- The **Piano Track** has a **MIDI target** reference to the **Piano** object on the Actor-Mixer Hierarchy.
- The **Drum Track** has a **MIDI target** reference to the **Drum Kit** object on the Actor-Mixer Hierarchy.
- The **Guitar Track** has no **MIDI target**.

When playing back a Music Segment with MIDI clips, the MIDI clip will send MIDI data to the specified MIDI target. The actual MIDI target (or instrument), located in the **Actor-Mixer Hierarchy** will receive the MIDI data, and play accordingly to its MIDI settings.

To specify a MIDI target on a Music object:

1. Select a Music object (Music Track, Music Segment or any parent).
2. In the **Property Editor**, go to the **MIDI** tab.
3. In the **MIDI Target** group, click **Override Parent** (if applicable).
4. Click the [...] button to browse a MIDI target instrument.

For more information, see [Chapter 26, Creating MIDI Instruments](#).

Mixing MIDI and Audio contents

When MIDI content is played-back in the Music Hierarchy, the music objects send MIDI data to the MIDI target (the instrument), located in the Actor-Mixer Hierarchy. The actual audio content played-back to the MIDI target will play in the Actor-Mixer Hierarchy context, not in the MIDI source location's context. This means the voice properties on music objects will not affect the instrument being played.

For example, the Voice Volume on a Music Segment or Music Track would not affect a piano instrument, located in Actor-Mixer Hierarchy. To modify the voice volume of the piano, you would need to modify the Voice volume on the Piano object directly, not on the MIDI source.

To help mixing, a bus hierarchy can be created to combine instrument content with music audio content. Routing the instrument and the music track to the same bus would provide a single audio mixing point.

Understanding MIDI Tempo

There is a major difference between a MIDI clip and a WAV clip. A WAV clip's duration is uniquely determined by the content of the source file. However, the duration of a MIDI file depends on the tempo used. The tempo used is determined upon the creation of the clip, via the Tempo Source property.

MIDI files normally contain the tempo information for playback at the right speed. This tempo information enters in conflict with the tempo information found in the Interactive Music objects (Segment and parents). You have the choice of selecting the tempo of the source MIDI files, or use the tempo of the Interactive Music Hierarchy.

The following table lists the possible tempo sources:

Tempo Source	Definition
File	The tempo contained in the MIDI file is used.
Hierarchy	The tempo specified by the MIDI clip's Interactive Music ancestry is used.

To set the MIDI tempo source:

1. Select a Music object.
2. In the **Property Editor**, go to the **MIDI** tab.
3. In the **MIDI Clip Tempo** group, select the **Source** to use.

A MIDI clip's tempo can be changed after creation, but the duration is not affected. The duration may be changed via the clip handles. For more information, refer to [Working with Clips](#).

Changing the Playback Speed of MIDI

The **Voice Pitch** property can't be set on the Interactive Music Hierarchy. However, the speed of playback in the Music Hierarchy can be changed.

Changing the **Playback Speed** of a Music object will:

- Affect the pitch of audio clips.
- Affect the rate of playback of MIDI clips.

A playback speed of 1 will playback at the original speed. A playback speed of 2 will play back twice as fast as the original. A playback speed of 0.5 will play at half speed.



Note

The **Playback Speed** property can also be attached to a Game Parameter in the RTPC tab of the **Property Editor**.

To change the Playback Speed:

1. Inspect a Music Object (except Music Track).
2. In the **Property Editor**, go to the **General Settings** tab.
3. Set the **Playback Speed**.

Chapter 26. Creating MIDI Instruments

Designing a Synth One Instrument	514
Designing a Simple Sampled MIDI Instrument	514
Understanding MIDI Note Tracking	515
Understanding the MIDI Filters	515
Understanding the MIDI Events	516
Adding Fade-in and Fade-out on MIDI events	517
Using MIDI Data to Control Object Property Values	517
Using the MIDI Keymap Editor	518
Testing an Instrument with a MIDI Keyboard	518
Routing MIDI from a DAW to Wwise	519

Designing a Synth One Instrument

It is possible to create a MIDI instrument from a Source Plug-in if the plug-in understands MIDI messages.

To create a source plug-in instrument (ex: synthesizer):

1. Create an empty **Sound** object on the **Actor-Mixer Hierarchy**.
2. In **Project Explorer**, select the **Sound**.
3. In the **Contents Editor**, click **Add Source >>**.
4. From the selector menu, select **Synth One**.
5. From the **Views** menu, select **Source Editor (Shift+X)**.
6. Click on the **Sound** object on the **Actor-Mixer Hierarchy** to see it in the **Source Editor**.
7. In the **Source Editor**, set the **Frequency Mode** to **MIDI Note**.

Now the instrument can be referenced as a MIDI target inside a music object (ex: Music Segment).

Designing a Simple Sampled MIDI Instrument

Sampled instruments can use all **Actor-Mixer Hierarchy** containers (Blend Containers, Switch Containers, Random Containers, Sequence Containers and Sounds) for their design. The complexity of the implementation will grow with the instrument design complexity. The most simple sampled instrument would be a 1-sample instrument.

To create a 1-sample instrument:

1. In the **Project Explorer**, select the location to create instrument.
2. From the **Views** menu, select **Audio File Importer (Shift+I)**.
3. Click **Add Files...**
4. Browse for a **.wav** file, and click **OK**.
5. Click **OK** again to complete the import operation.



Tip

You can also Drag & Drop a WAV file on the **Actor-Mixer Hierarchy** directly.

Now the instrument can be referenced as a MIDI target inside a music object (ex: Music Segment).



Tip

Use the **Source Editor** to trim the imported samples. You normally want to remove any silence at the beginning of the source.

Understanding MIDI Note Tracking

The MIDI note tracking parameters can be found in the **MIDI tab** of the **Property Editor** for the **Actor-Mixer Hierarchy** objects. These parameters determine if, upon reception of a MIDI message, the sound objects are pitch-shifted during playback. If the played sound is pitch-shifted, it is according to the note of the MIDI message, and the note represented by the Actor-Mixer object's source; it's root note.

The note tracking parameters can be specified and/or overridden in any of the instrument's **Actor-Mixer Hierarchy** objects. For more information about properties in the Actor-Mixer hierarchy, refer to [About Properties in the Project Hierarchy](#). The MIDI note tracking parameters are:

- **Override parent:** if set, the object ignores the note tracking parameters of its ancestor objects.
- **Enable:** if set, the object's sound is pitch-shifted when played. The pitch-shift is done using the received MIDI message's note and the parameter **Root note**.
- **Root note:** the note represented by the object's source.

Understanding the MIDI Filters

The MIDI filters can be found in the **MIDI tab** of the **Property Editor** for the **Actor-Mixer Hierarchy** objects. When receiving a MIDI message on a complex object structure, you can use the MIDI filters to select which child object to play.

The MIDI filters define which child object to play will based on:

- MIDI note key.
- MIDI note velocity.
- MIDI channel.

Because re-sampling has a tendency to create unwanted artifacts as sounds are pitched away from their root notes, you may want to use multiple recorded samples at different root notes to cover the full extent of an instrument. A technique often used that usually provides good results when memory is limited is to use a few different sample root notes per octave and pitch them up and down to cover the full 12 semitones of an octave.

Here's an example for two octaves of an instrument pitching the root notes down by a second and up by a minor second:

- **Instrument (Blend Container)**
 - Root note: C3 - Range: Bb2 to C#3
 - Root note: E3 - Range: D3 to F3
 - Root note: G#3 - Range: F#3 to A3
 - Root note: C4 - Range: Bb3 to C#4
 - Root note: E4 - Range: D4 to F4
 - Root note: G#4 - Range: F#4 to A4

The Blend Container simultaneously plays every child when playing the container. The filter however blocks the children that don't match the filter rules.

Understanding the MIDI Events

The MIDI Events properties can be found in the **MIDI** tab of the **Property Editor** for the **Actor-Mixer Hierarchy** objects. When receiving a MIDI message, the MIDI Events properties are used to decide whether the object must be played. The object can be played on either a note-on or note-off event. Note that these properties are only used to start the playback of the object. To stop the object playback, an envelope must be assigned to a property (refer to [Working with Envelopes](#)).

A typical scenario would be to **Play on Note-On**.

To create a looping instrument:

- In the **Project Explorer**, select the Sound to loop.
- In the **Property Editor**, go to the **General Settings** tab.
- Enable **Loop** on the Sound.
- In the **Property Editor**, go to the **MIDI** tab.
- Set the **Play On** property to **Note-On**.
- From the **Views** menu, open the **Source Editor** (Shift+X).
- Inspect the Sound again.
- In the **Source Editor**, move the **Loop Start** and **Loop End** cursors to exclude the attack and release sections of the WAV file.
- Adjust the **Crossfade duration** until the loop point can't be heard.



Note

The instrument's looping may be stopped via the **Break on Note-Off** property. If the property is set, the note-off stops the

playback of the looped sound while allowing the current object to finish playing. Remember to set the **Loop Start** and **Loop End** cursors in the source editor such that the final loop plays for the desired duration.

Another scenario would be to have a specific sound that is played for the release of an instrument, at Note-Off. For example, this could be used to trigger the string muting sound of a guitar note ending. You would create the following objects and settings:

- Blend Container
 - Attack+Loop Sound: Play On = Note-On
 - Release Sound: Play On = Note-Off

Adding Fade-in and Fade-out on MIDI events

When you want to add dynamics to your instruments, you can use an **Envelope** attached to the **Voice Volume** of the instrument.

To add an **Envelope** to the **Voice Volume**, refer to [Working with Envelopes](#).

Using MIDI Data to Control Object Property Values

It is possible to use the following MIDI messages to control object property values:

- MIDI Note Velocity.
- MIDI Note Key (number).
- MIDI Note Frequency.
- MIDI Note Aftertouch.
- MIDI CC values (0-127, including Modulation Wheel).
- MIDI Pitch bend.

A typical scenario would be to control the **Voice Volume** of the Instrument with the **MIDI Note Velocity**.

To control the Voice Volume with the MIDI Note Velocity:

1. In **Project Explorer**, select an object from the Actor-Mixer Hierarchy.
2. In the **Property Editor**, go to the RTPC tab.
3. Click the [**>>**] button in the RTPC list to add a new entry.
4. From the selector menu, select **Voice Volume**.
5. Click the [**>>**] selector for the X axis.
6. From the selector menu, select **MIDI > MIDI Note Velocity**.

7. Adjust the Voice Volume curve in the RTPC graph.

Using the MIDI Keymap Editor

The MIDI Keymap Editor view can be used to edit all MIDI properties for Actor-Mixer Hierarchy objects.

To open the view:

1. In **Project Explorer**, select an object from the Actor-Mixer Hierarchy.
2. In **Property Editor**, go to MIDI tab.
3. Press the **Keymap Editor** button.

To set the same property values on multiple objects:

1. Select the objects to edit in the **MIDI Keymap Editor**.
2. Set the value of a property on one of the selected objects.

The selected objects are now set to the same value.

To offset the property values of multiple objects:

1. Select the objects to edit in the **MIDI Keymap Editor**.
2. Hold the ALT key and move the property slider on one of the selected objects.

The selected objects have their property value offset.

To add properties in the MIDI Keymap Editor:

1. Open the **MIDI Keymap Editor** view settings (Ctrl+Alt+V).
2. Select the properties to add.
3. Press OK.

New columns are added.

Testing an Instrument with a MIDI Keyboard

When designing a MIDI instrument, you can test the instrument with an external MIDI keyboard device.

To connect your device to Wwise:

1. From the **Project** menu, select **Control Surface Devices**.
2. Click the **Add** button.
3. Give a name to your device.

4. Click OK.

The device is added to the list.

5. In the **Receive From** column, select the MIDI IN device.

The **Connected** message appears.

6. In the **Send To** column, select the MIDI OUT device.

The **Connected** message appears.

7. Click Close.

The device is now ready to use.

To bind the keyboard keys to the current selection:

1. From the **Views** menu, select **Control Surface Bindings** (Ctrl+Shift+Q).
2. Create a new Control Surface Session by clicking the [>>] button on the top left of the view.
3. Click the **Current Selection** group (folder).
4. Click the **Add Binding** button.
5. Click the **Property/Command** selector button to open the menu.
6. Select **Object Command > Pass MIDI Note**.
7. Save your project.



Note

Ensure the **Current Selection** group is active in the Control Surface toolbar and that you have an object selected in **Project Explorer**.

After creating the session and the binding, selecting an object in the Project Explorer will automatically load the object for the MIDI instrument, ready to play.

Routing MIDI from a DAW to Wwise

It is possible to route MIDI messages coming from an external application (DAW, or Digital Audio Workstation) to Wwise using a virtual MIDI connector. This can be useful to compose MIDI music in a DAW while using the instruments built in Wwise.

You can author multiple instruments simultaneously by creating a **Blend Container** containing all instruments, and using the **MIDI Channel** filter to discriminate the instruments.

Tobias Erichsen's loopMIDI is an example application that can be used to route MIDI message from a DAW to Wwise. You will need to add the virtual MIDI ports created by the application in the Wwise's Control Surface Devices.

Chapter 27. Working with Transitions

Overview	522
Understanding Transitions	523
Adding Transitions	524
Copying and Pasting Transitions	525
Removing Transitions	526
Setting Source and Destination Properties	526
Using Transition Segments	530
Interactive Music Transitions Tips & Best Practices	532

Overview

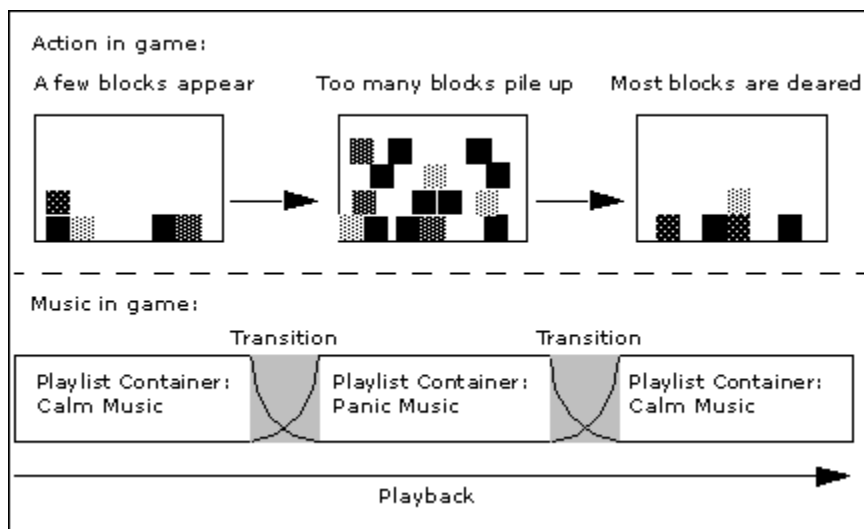
Transitions are the key to making interactive music listenable. Without smooth transitions, the illusion of a game score being one constantly-shifting piece of music is lost. Individual music segments fitted together badly are jarring for the listener and break the realism of your game. To avoid these problems, Wwise allows you to customize the transitions between music objects and make them as seamless as possible.

A transition is meant to be a smooth bridge between a source and destination segment, and Wwise provides you with tools to refine these bridges. You can define transitions between individual segments, or between containers holding segments. In this way, you can avoid discordant and sudden changes between musical passages.

Using Transitions - Example

Imagine you are making a block puzzle game. At any given time, your players experience one of two situations: the game is going well (blocks are being cleared rapidly) or the game is going badly (blocks are piling up). Naturally, you've had some cool music composed to reflect each of these situations, which you want to show off to its fullest. One way to do this is to make sure the transitions between these two sets of music sound natural and musical.

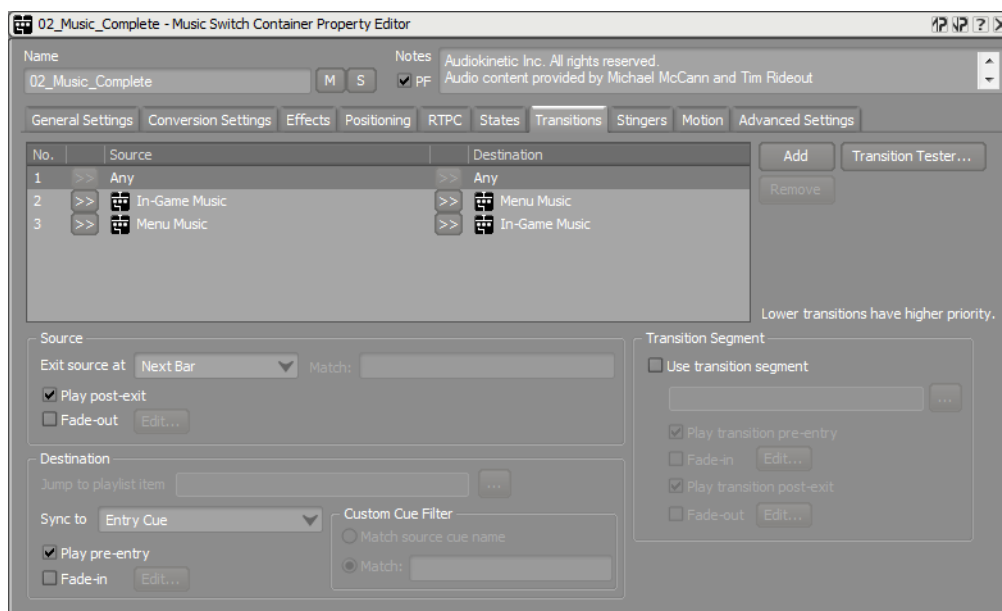
To meet the needs of your game, you could create a music switch container called Puzzle Sounds, with two music playlist containers, Calm and Panic. These containers would be linked to two game states, reflecting the in-game situations. Successful transitions between these two playlist containers would make the game's score compelling. The following illustration shows how these transitions could be implemented in game:



In this case, the transitions between the playlist containers have been smoothed by the use of fade-in and fade-out curves. This makes the change from one container to another more gradual and natural, and avoids noisy artifacts.

Understanding Transitions

An interactive music transition is a musical behavior that occurs whenever one music object, called a source, stops playing and another, called a destination, begins. Each transition is carried out automatically by Wwise according to properties you define in the **Transitions** tab of the **Property Editor**.



Wwise implements a transition whenever your game music changes from one interactive music segment to another, or from one container holding segments to another. You can specify when a transition will take place and if fades will accompany it. You can also assign a musical passage called a transition segment to play during a transition. Overall, you can customize your transitions in Wwise to make them a seamless part of your interactive music.

Understanding the Transition Matrix

The center of the **Transitions** tab is the Transition Matrix, a list of rules that defines how each object within a music switch or playlist container transitions to every other object within the container. When working with transitions, you can create explicit rules for each object in a container, or more general rules that apply to more than one object. You can define any one of the following as the source or destination of a transition rule:

- **Music object:** This can be a segment, a music playlist container, or a music switch container.

- **Virtual Folder:** Music objects that are within a music switch container can be organized into virtual folders. When a virtual folder is selected as a source or destination for a music transitions, the transition rule applies to any of the music objects within the folder.
- **Any:** This means that any music object in the container can be used as the source or destination.
- **Nothing:** This means that either the source or destination is empty, that is, there is no music object playing.

When a transition is needed, Wwise begins searching at the bottom of the list until it finds one that applies to the current situation. If no matching transition is found, Wwise uses the default “Any to Any” transition.

Adding Transitions

The Transition Matrix for a music switch or playlist container always contains at least one transition: Any to Any. If you want to create more specific transition rules, you must add them to the Transition Matrix yourself.

To add a defined transition to a music switch or playlist container:

1. Load a music switch or playlist container into the Property Editor.
2. Switch to the Transitions tab.
3. Click **Add**.

A new Any to Any transition is added to the Transition Matrix.

4. In the Source column, click the Selector button (>>) and select one of the following options:

Any to create a transition for any unspecified source music object.

Nothing to create a transition where there is no source, that is, no music object is playing.

Browse to create a transition using a specific music object, or a group of objects that are inside a virtual folder, as the source. The **Project Explorer - Browser** opens where you can select a source.



Tip

You can also drag music objects directly from the **Project Explorer** to the Transition Matrix. It is important to note, however, that only music objects that are children of the

container loaded into the **Property Editor** can be added to the matrix.

5. In the **Destination** column, click the **Selector (>>)** and select one of the following options:

Any to create a transition for any unspecified destination music object.

Nothing to create a transition where there is no destination, that is, no music object is playing.

Browse to create a transition using a specific music object, or a group of objects that are inside a virtual folder, as the destination. The **Project Explorer - Browser** opens where you can select a destination.

The new customized transition is added to the Transition Matrix.



Tip

You can change the order of transitions in the Transition Matrix by selecting them and dragging them up and down. A red line shows you where the transition will be placed. The “Any to Any” transition will always be the last one Wwise checks, and therefore cannot be moved from the top of the Matrix.

Related Topics

- [Removing Transitions](#)
- [Setting Source and Destination Properties](#)
- [Copying and Pasting Transitions](#)
- [Using Transition Segments](#)

Copying and Pasting Transitions

To speed up the workflow of creating transitions, you can easily copy and paste transition rules within the matrix.

To copy a defined transition from a music switch or playlist container:

1. Load a music switch or playlist container into the Property Editor.
2. Switch to the Transitions tab.
3. In the Transition Matrix, right-click the transition you want to copy and select **Copy** from the menu.

4. Right-click where you want the transition to be appear in the list, and select **Paste** from the menu.

A copy of the transition is added to the Transition Matrix in the selected location.



Tip

You can also use the shortcut keys (Ctrl+C, Ctrl+V) to copy and paste transitions within the matrix.

Related Topics

- [Adding Transitions](#)
- [Setting Source and Destination Properties](#)
- [Using Transition Segments](#)

Removing Transitions

If you no longer require a transition rule you have defined in the Transition Matrix, you can easily remove it. The source and destination music objects are not deleted when you remove a transition rule.

To remove a defined transition from a music switch or playlist container:

1. Load a music switch or playlist container into the **Property Editor**.
2. Switch to the **Transitions** tab.
3. In the Transition Matrix, select the transition you want to remove.
4. Click **Remove** or press the **Delete** key.

The selected transition is removed from the Transition Matrix.

Related Topics

- [Adding Transitions](#)
- [Setting Source and Destination Properties](#)
- [Copying and Pasting Transitions](#)
- [Using Transition Segments](#)

Setting Source and Destination Properties

By default, a transition is a simple change from one music object to another. The real power of transitions appears when you customize the source and

destination to make a unique musical passage. By setting source and destination properties, you can make a transition between objects sound both smooth and musical.

To give you additional control and flexibility over the transitions between objects, you can choose from a number of different exit and entry points for the source and destination respectively. You can also decide to pick random entry points in the destination object. This results in a different entry point being used each time the transition is played making your transitions less repetitive.

When you set the properties for a source or destination, keep in mind that you are setting them for that one transition only. For example, let's say your transition matrix defines transitions from Happy Music to Sad Music and from Happy Music to Scary Music. If you want each Happy Music source to exit at the first custom cue, you will have to set this property separately for each transition.



Note

Many different properties can be set for each source and destination, though the properties available will depend on the type of object being used and the type of container holding it.

To set the properties of a source:

1. Load a music switch or playlist container into the Property Editor.
2. Switch to the **Transitions** tab.
3. Select a transition from the Transition Matrix.

You can now edit the properties of the source for this transition.

4. If the source is in a switch container, select one of the following options from the **Exit source at list**:

Immediate: The source stops playing immediately.

Next Grid: The source stops playing at the next grid interval. The grid is an arbitrary method by which music objects can be virtually partitioned.

Next Bar: The source stops playing at the next bar.

Next Beat: The source stops playing at the next beat.

Next Cue: The source stops playing at the next cue, whether it be a custom cue or the exit cue.

Next Custom Cue: The source stops playing at the next custom cue. If the current music segment doesn't contain a custom cue, Wwise continues to the next segment until it finds a custom cue.

Exit Cue: The source stops playing at the exit cue.

5. If you selected Next Cue or Next Custom Cue, you may further refine the selection of valid cues where the transition may occur by entering a cue name in the **Match** edit box.
6. If you want the post-exit of the source to play during the transition, select **Play post-exit**.



Note

The post-exit of a source will only play if that source exits at its exit cue, or fades out at or beyond its exit cue. Otherwise, the post-exit will never play during a transition.

7. If you want the source to end playing with a fade-out, select **Fade-out**.

To set the properties of a destination:

1. Load a music switch or playlist container into the Property Editor.
2. Switch to the **Transitions** tab.
3. Select a transition from the Transition Matrix.

You can now edit the properties of the destination for this transition.

4. If the destination is a music playlist container, you can select a specific item within the playlist container to be played first. Click the **Browse** button (...) to specify the item for the **Jump to playlist item** option.
5. If the owner of the transition is a switch container, select one of the following options from the **Sync to list**:

Entry Cue: The destination will begin playing at its entry cue.

Same Time as Playing Segment: The destination will begin playing at the same time mark as the source segment. For example, if the source segment has been playing for 10 seconds since its beginning, the destination will start playing 10 seconds in.

Random Cue: The destination will begin playing at a randomly chosen cue. When this option is selected, any cue can be chosen, including the entry cue or any custom cue.

Random Custom Cue: The destination will begin playing at a randomly chosen custom cue. If there are no custom cues in the segment, the entry cue will be used.

6. If you selected Random Cue or Random Custom Cue, you may further refine the selection of valid start positions with the Custom Cue Filter.

Match: Only cues with this name may be selected as the start position.

Match source cue name: Only cues which name is the same as the name of the cue that was used in the source segment for this transition may be selected as the start position.

7. If you want the pre-entry of the destination to play during the transition, select **Play pre-entry**.
8. If you want the destination to begin playing with a fade-in, select **Fade-in**.

Related Topics

- [Adding Transitions](#)
- [Removing Transitions](#)
- [Copying and Pasting Transitions](#)
- [Using Transition Segments](#)

Editing Fades

Fade-ins and fade-outs are special properties assigned to destination and source music objects, respectively. They can also be applied to the beginning and endings of transition segments. By using fades, you can refine the entrance and exit of music objects and establish smoother transitions. You can define the length and offset of each fade, as well as the curve shape to further customize its sound.

To edit a fade-out:

1. On the Transitions tab of the Property Editor, make sure **Fade-out** is selected, and then click **Edit**.

The **Music Fade Editor** opens.

2. In the **Time** field, specify the duration of the fade-out.
3. In the **Offset** field, specify the amount of time between the exit cue and the end of the fade-out.
4. From the **Curve** list, select a curve shape for the fade-out.

The fade-out will be played according to your specifications.

To edit a fade-in:

1. On the **Transitions** tab of the **Property Editor**, make sure **Fade-in** is selected, and then click **Edit**.

The **Music Fade Editor** opens.

2. In the **Time** field, specify the duration of the fade-in.
3. In the **Offset** field, specify the amount of time between the entry cue and the beginning of the fade-in.
4. From the **Curve** list, select a curve shape for the fade-in.

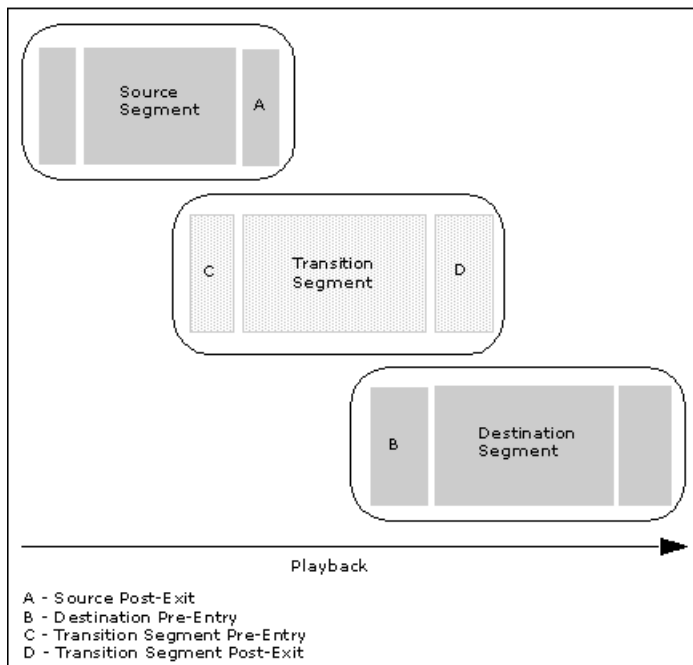
The fade-in will be played according to your specifications.

Related Topics

- [Adding Transitions](#)
- [Removing Transitions](#)
- [Setting Source and Destination Properties](#)
- [Copying and Pasting Transitions](#)
- [Using Transition Segments](#)

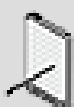
Using Transition Segments

At times, a transition sounds better if another piece of music plays over the end of the source and the beginning of the destination. This bridging piece of music is called a transition segment, and you can use one for any transition in Wwise. The following illustration shows how a transition segment is played between the source and destination of a music transition.



You can also use any combination of the pre-entry and post exit areas of the source, destination, and transition segments to create even more seamless transitions.

Any music segment can be used as a transition segment. To learn how to create a segment, refer to [Chapter 24, Working with Music Tracks and Segments](#).



Note

If a work unit containing the transition segment has been unloaded from the project, the transition segment will be highlighted in yellow.

To use a transition segment:

1. Load a music switch or playlist container into the **Property Editor**.
2. Switch to the Transitions tab.
3. Select a transition in the Transition Matrix.
4. Select **Use transition segment**.
5. Do one of the following:

Drag a segment from the **Project Explorer** to the **Transition Segment** field.



Click the **Browse** button (...) and select a segment from the **Project Explorer - Browser**.

6. If you want the pre-entry of the transition segment to play, select **Play transition pre-entry**.
7. If you want the transition to begin playing with a fade-in, select **Fade-in**.
8. If you want the post-exit of the transition segment to play, select **Play transition post-exit**.
9. If you want the transition to stop playing with a fade-out, select **Fade-out**.



Note

For more information about editing the fade-in and fade-out properties of a transition segment, refer to [Editing Fades](#).

Related Topics

- [Adding Transitions](#)
- [Removing Transitions](#)
- [Copying and Pasting Transitions](#)
- [Setting Source and Destination Properties](#)

Interactive Music Transitions Tips & Best Practices

Interactive music is a complex tool with many options. Adopting a coherent strategy towards interactive music at the beginning of a project can save you time and effort later on. Of course, there are multiple ways to approach any interactive music project, and you can use Wwise in any way you see fit to create the best results for your game. The following are suggestions for how you can better manage your interactive music transitions.

Ordering Transitions

When a transition is required, Wwise scans the Transition Matrix from bottom to top to find a transition rule that matches the current situation. It will stop as soon as it finds a transition that fits, whether or not this is the best transition to apply. To make sure your transitions are applied optimally, arrange the rules you have defined in the Transition Matrix in descending order, from general to specific, as in the following example.

Source		Destination	
>>	Any	>>	Any
>>	Any	>>	Exciting Music
>>	Exciting Music	>>	Any
>>	Casual Music	>>	Creepy Music
>>	Creepy Music	>>	Casual Music

Transitions with **Any** or **Nothing** as their source or destination should appear higher on the list than those naming specific music objects. If you do this, Wwise will encounter the specific rules you have defined before any general ones. For more information on creating and arranging transitions, refer to [Adding Transitions](#).

Chapter 28. Using Stingers

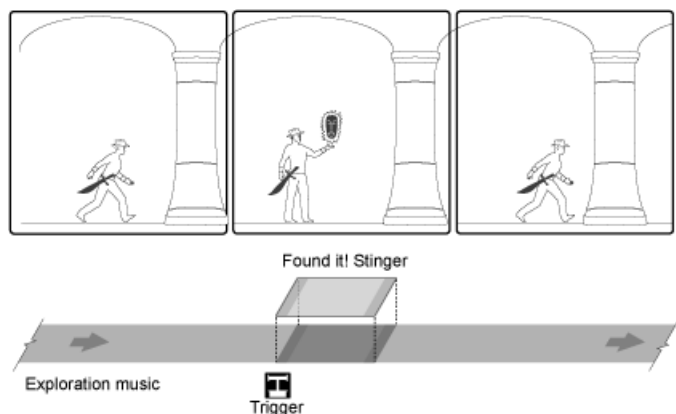
Overview	535
Adding Stingers	536
Defining Playback Settings for Stingers	538
Removing Stingers	539
Auditioning Stingers	540

Overview

To add more feedback to your interactive music, you can play stingers at key points in the game action. Stingers are brief musical phrases that are superimposed and mixed over the currently playing music. To play a stinger, the game calls a trigger that is associated with a stinger music segment. For more information on using triggers, refer to [Chapter 19, Working with Triggers](#).

Using Stingers- Example

Let's say the dashing archeologist that we met in the previous example is busily exploring a temple in search of artifacts and adventure. The exploration music plays as he wanders about examining his surroundings. Then as he finds the treasure, another short piece of music plays to emphasize this exciting discovery. The exploration music is not replaced, but rather the game calls a trigger which in turn plays a stinger called "Found it!" over the ongoing exploration music to signify this big moment. After the stinger has finished, the exploration music continues to play.



To insert a stinger on top of the already playing music, you need to do the following:

- Associate a music object to a trigger.
- Create the stinger by mapping a music segment to a trigger.
- Define how the stinger will play back.

Using Stingers in the Interactive Music Hierarchy

Since stingers can be created at different levels in the hierarchy, you can assign the same trigger to different segments. This means that while the top level music object may use a trigger called, for example, "Headshot", any of its child objects may also use the "Headshot" trigger. In this case, the child object would associate the "Headshot" trigger with a different segment. This automatically overrides the parent trigger/stinger association and increases the range of

stingers that you can play at important moments in the game. Since only one stinger may play for a specific trigger in the hierarchy of music objects, it is the associated stinger of the currently playing child object that will play.

Working with stingers includes the following tasks:

- [Adding Stingers](#)
- [Defining Playback Settings for Stingers](#)
- [Auditioning Stingers](#)
- [Removing Stingers](#)

Adding Stingers

You can create stingers for music objects in the Stingers tab of the Music Object Property Editor. In Wwise, there are two ways to add stingers:

- Using drag and drop from the Project Explorer.
- Using the buttons in the Property Editor.

You can use either method or a combination of the two.



Note

If your stinger contains a segment that has been unloaded from the project, the segment will be highlighted in yellow.

To insert a stinger using drag and drop from the Project Explorer:

1. Load a music object into the Property Editor and switch to the **Stingers** tab.
2. From the Game Syncs tab of the Project Explorer, drag a trigger to the stingers list.

A stinger is automatically created using the trigger you dragged from the Project Explorer. By default, the trigger is not associated with any music segment (Nothing).

3. To assign a music segment to this trigger, switch to the Audio tab of the Project Explorer, and then drag the music segment to the Segment to Play column.

This music segment replaces the “Nothing” option that is added by default, and is now associated with the trigger.

4. In the Play At column, define when you want the segment to play by selecting one of the following options:

Immediate - Change occurs immediately. If a look-ahead time has been defined for a track, then this time must elapse before the stinger can play.

Next Grid - Stinger plays at the next grid. The grid is an arbitrary frequency by which music objects can be virtually partitioned.

Next Bar - Stinger plays at the next bar.

Next Beat - Stinger plays at the next beat.

Next Cue - Stinger plays at the next cue. The next cue could be an Entry, Exit, or custom cue.

Next Custom Cue - Stinger plays at the next custom cue.

Entry Cue - Stinger plays at the Entry cue.

Exit Cue - Stinger plays at the Exit cue.

5. Repeat steps 2-4 to continue adding stingers as needed.

To insert a stinger using the buttons in the Property Editor:

1. On the Stingers tab of the Property Editor, click **Add**.

The Project Explorer - Browser dialog box opens.

2. Select the trigger to assign to the music object, and click **OK**.



Note

To select another trigger for the stinger, in the Trigger column click the **Browse** button (...) to browse to the trigger you want to use for this stinger.

3. Select the trigger, and click **OK**

The trigger is added to the stinger.

4. In the Segment to Play column, click the **Browse** button (...).

The Project Explorer-Browser opens.

5. Navigate to the segment that you want to use, and click **OK**.

The segment is added to the stinger.

6. In the Play At column, define when you want the segment to play by selecting one of the following options:

Immediate - Change occurs immediately. If a look-ahead time has been defined for a track, then this time must elapse before the stinger can play.

Next Grid - Stinger plays at the next grid. The grid is an arbitrary frequency by which music objects can be virtually partitioned.

Next Bar - Stinger plays at the next bar.

Next Beat - Stinger plays at the next beat.

Next Cue - Stinger plays at the next cue. The next cue could be an Entry, Exit, or custom cue.

Next Custom Cue - Stinger plays at the next custom cue.

Entry Cue - Stinger plays at the Entry cue.

Exit Cue - Stinger plays at the Exit cue.

7. Continue to add stingers as needed.

Related Topics

- [Defining Playback Settings for Stingers](#)
- [Removing Stingers](#)
- [Auditioning Stingers](#)

Defining Playback Settings for Stingers

After you have created the stingers for a music object, you can define certain settings that help manage the playback of stingers in game. To obtain the best results for using a stinger you need to consider two playback issues:

- **The time you want to elapse before a stinger plays again.** Since stingers are meant to add intensity to a music score, it is important to avoid diluting this impact by playing a stinger too often and soon after one has played.
- **What to do when there is not enough time left in a segment to launch a stinger.** If a stinger is called by a trigger, and the next opportunity to play the stinger occurs too late in the current segment, the stinger is not played. You can, however, play the segment at the first defined opportunity of the next segment in the playlist. If you do not select this option the stinger will be killed.

To define the playback settings for your stingers:

1. On the Stingers tab of the Property Editor, in the **Don't play this stinger again for x seconds** field, type the number of seconds that must elapse before the stinger can play again. If the trigger calls this stinger within the specified number of seconds, the trigger is ignored.



Note

The "don't play stinger again for" time is relative to the stinger's synchronization point, that is, the moment in time where the stinger's entry cue occurs.

2. To play a stinger in the next segment when there is not enough time to play it in the current segment, select the **Allow playing the stinger in next segment** option.



Note

If you do not select this option, the stinger will be killed.

Related Topics

- [Adding Stingers](#)
- [Removing Stingers](#)
- [Auditioning Stingers](#)

Removing Stingers

You can delete any stingers that you no longer need.

To remove a stinger from the stinger list:

1. In the Stingers tab, select the stinger that you want to remove.
2. Click **Remove**.

The selected stinger is removed from the list.



Note

When a stinger is deleted, this does not affect the associated trigger or music segment.

Related Topics

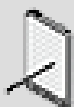
- [Defining Playback Settings for Stingers](#)
- [Adding Stingers](#)
- [Auditioning Stingers](#)

Auditioning Stingers

After you have created a stinger segment, you can audition it in the Transport Control on its own or with another segment to verify its impact when it plays over other music. Since stingers can be created at different levels in the hierarchy, you may have different segment settings that use the same trigger. Only one stinger, however, may play for a specific trigger in the hierarchy of music objects and it is the child object that determines which stinger will play. For example if you created a stinger using the trigger “Headshot” at the music switch container level, and used the same trigger “Headshot” for a stinger in the child object playlist container, and then you loaded the music switch container in the Transport Control, and called the trigger, only the playlist container stinger would play.

In addition, you can also create simulations in the Soundcaster and profile your simulation in the Profiler to monitor performance issues. For more information about simulations and profiling, refer to:

- [Building a Simulation](#)
- [Monitoring and Troubleshooting with the Performance Monitor](#)



Note

If a stinger contains a segment that has been unloaded from the project, the segment will be highlighted in yellow. You will not be able to audition your stinger until the work unit that contains the stinger segment has been loaded back into the project.

To audition a stinger in the Transport Control:

1. Load the stinger segment into the Transport Control.



2. Click the **Play** icon.

The stinger segment will play back.

To audition a stinger superimposed over currently playing music:

1. Load a music object into the Transport Control.
2. Click the **Play** icon.

The music object loaded into the Transport Control will play back.

3. In the Game Syncs area, click the **Triggers** button to display the trigger list.



4. Click the **Call Trigger** icon.

The corresponding stinger will play over the currently playing music object. You can continue to select other triggers and play back their corresponding stingers to simulate the playback of stingers in game.

Related Topics

- [Defining Playback Settings for Stingers](#)
- [Adding Stingers](#)
- [Removing Stingers](#)

audio**kinetic**

Part VI. Finishing Your Project



29. Managing Output	545
Overview	546
Specifying the Output Routing for Sound, Music, and Motion Objects	547
Using the User-Defined Auxiliary Sends	549
Using the Game-Defined Auxiliary Sends	550
Using Loudness Normalization or Make-up gain to Adjust Volume.....	551
Understanding the Voice Pipeline	552
Understanding Secondary Outputs	554
Understanding HDR	558
Using HDR	562
More about HDR	571
Creating the Final Mix	576
30. Managing Platform and Language Versions	594
Overview	595
Authoring Across Platforms	595
Localizing Your Project	620
Versions Tips and Best Practices	628
31. Creating Simulations	631
Overview	632
Building a Simulation	633
Managing Playback of Your Simulation	637
Simulating with Game Syncs	640
Fine-Tuning Properties in a Simulation	644
Creating Simulations Tips and Best Practices	648
32. Managing Memory in Wwise	649
Overview	650
Understanding the Components of the Memory Manager	650
Setting the Size of Your Memory Pools	651
Troubleshooting Memory Problems	654
Optimizing Memory Pools	654
Memory Management Tips and Best Practices	660
33. Profiling	662
Overview	663
Understanding the Different Types of Profiling in Wwise	664
Connecting to a Local/Remote PC or Game Console	669
Capturing Data from the Sound Engine	672
Monitoring and Troubleshooting with the Performance Monitor	683
Keeping Track of Objects and Listeners with the Game Object Explorer	686
Examining Objects with the Game Object 3D Viewer	691
Evaluating Game Syncs with the Game Sync Monitor	696
Profiling Tips and Best Practices	697
34. Managing SoundBanks	699
Overview	700

Understanding How SoundBanks are Loaded in a Game	702
Building SoundBanks	707
Managing SoundBanks	725
Defining Custom Attributes for Your SoundBanks	728
Generating SoundBanks for a Project	738
Using the CopyStreamedFiles Tool	743
Strategies for Managing SoundBanks	744
SoundBanks Tips and Best Practices	759
35. Managing File Packages	761
Overview	762
Working with File Packager Projects	762
Managing File Packages within a Project	765
Downloadable Content Overview	770
Generating File Packages	772
Using File Packager Arguments in the Command Line	773
File Packager Tips and Best Practices	777

Chapter 29. Managing Output

Overview	546
Specifying the Output Routing for Sound, Music, and Motion Objects	547
Using the User-Defined Auxiliary Sends	549
Using the Game-Defined Auxiliary Sends	550
Using Loudness Normalization or Make-up gain to Adjust Volume	551
Understanding the Voice Pipeline	552
Understanding Secondary Outputs	554
Understanding HDR	558
Using HDR	562
More about HDR	571
Creating the Final Mix	576

Overview

On top of the project hierarchy that includes all your sound, music, and motion objects sits the **Master-Mixer Hierarchy**. This separate hierarchical structure of busses - described earlier in [Chapter 8, *Building the Structure of Output Busses*](#) - allows you to group the many different sound, music, and motion structures within your project hierarchy. It is divided into three main branches with a top-level master bus:

- [The Master Audio Bus Hierarchy](#), for the main sound mix
- [The Master Secondary Bus Hierarchy](#), for any secondary sound mixes you may have, such as to a game controller
- [The Master Motion Bus Hierarchy](#), for motion sound mixing.

These master busses create the final level of control for the sound, music, and motion structures within your project. Because busses sit on top of your project hierarchy, you can use them to create the final mix for your game.

Depending on the platform, you can also apply certain effects to the Master Audio and Master Secondary busses to help finalize that immersive experience for your game.



Note

Motion busses cannot have effects.

Related Topics

- [Overview](#)
- [Defining the Properties of a Bus](#)
- [Structuring a Bus Hierarchy - Example](#)

Audio Output Formats

Wwise outputs the final mix as a discrete number of LPCM channels. It is important to note that Wwise does not do any multi-channel encoding; it simply feeds LPCM data to the console or system in either stereo or 5.1 surround. Once the LPCM data is received by the console or system, it can then be output in almost any format supported by the particular console or system, including Dolby, DTS, or DPL2.



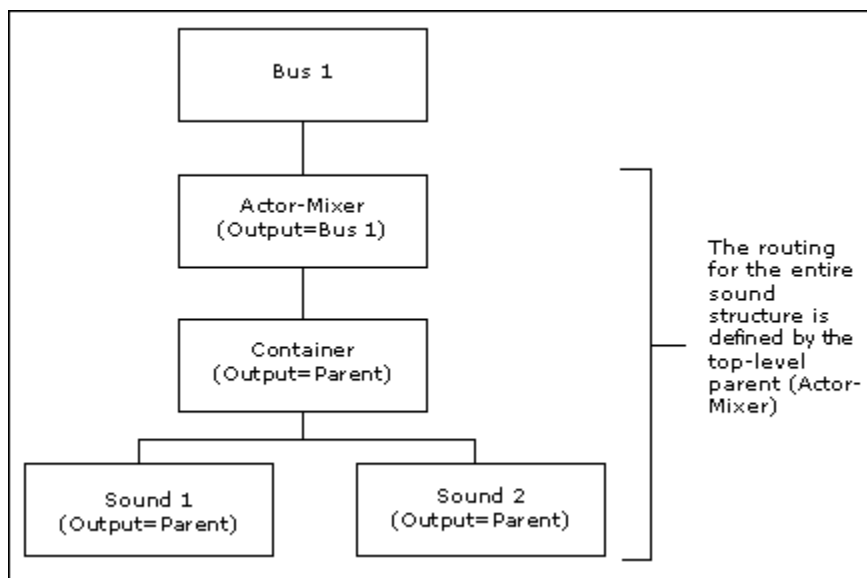
Note

Wwise does not support 7.1 surround on the PlayStation 3.

Specifying the Output Routing for Sound, Music, and Motion Objects

Each object within the hierarchy must be routed to a specific bus. You can define the routing for an entire sound or motion structure simply by setting them for the top-level parent object. Since the output routing is an absolute property, these settings will automatically be passed down to the child objects below it. You can, however, override the parent settings, if necessary.

The following illustration shows how the output routing set for a parent object is automatically passed to its child objects.

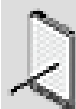


Note

By default sound, music, and motion FX objects, are automatically routed to the busses specified for Motion and Audio Output in the Default User Settings dialog box.

To specify the output routing for sound and music objects:

1. Load a top-level object into the Property Editor.



Note

If the object is not a top-level object, you must select the Override parent option before you can set the Output options.

2. In the Audio Output Bus group box, click the **Browse** button (...).

The Project Explorer - Browser opens.

3. Select the audio bus through which you want the object to be routed.
4. Click **OK**.

The audio bus is applied to the current object and any child objects below it.

To unlink the output routing for the current platform:

1. Load a top-level object into the Property Editor.



Note

If the object is not a top-level object, you must select the **Override parent** option before you can set the Output options.

2. In the Audio Output Bus group box, right-click the current bus name to open the shortcut menu.
3. Select *Unlink* from the shortcut menu.

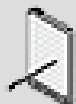
Now the bus routing is unlinked and you can specify another bus for this platform.

Routing for Motion Objects

Since motion objects have a different bus hierarchy than sound and music objects, you have to set the routing for motion objects separately. Even if you plan to generate motion from existing sounds and music objects, you have to specify the routing separately for the motion data. For more information on generating motion from existing sounds, refer to [Generating Motion from Existing Sounds](#).

To specify the output routing for motion objects:

1. Load a top-level object into the Property Editor.



Note

If the object is not a top-level object, you must select the **Override parent** option before you can set the Output options.

2. In the Motion Output Bus group box, click the **Browse** button (...).

The Project Explorer - Browser opens.

3. Select the motion bus through which you want the object to be routed.
4. Click OK.

The motion bus is applied to the current object and any child objects below it.

Related Topics

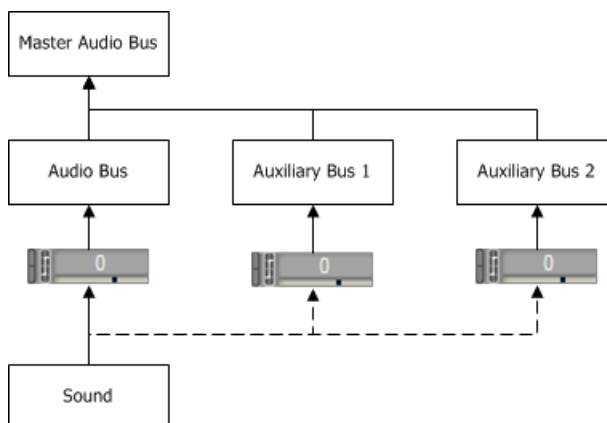
- [Defining Absolute Properties](#)
- [Applying Effects to Objects](#)
- [Overriding Parent Properties](#)
- [Defining Relative Properties \(Volume, Pitch, LPF, HPF\)](#)

Using the User-Defined Auxiliary Sends

Additionally to the audio bus specified for the audio routing, every actor-mixer object or interactive music object can specify up to 4 user-defined auxiliary sends. An auxiliary send allows to send a portion of the audio signal to an additional bus for parallel processing.

User-defined auxiliary sends are used to define static auxiliary sends, directly in the authoring, as opposed to game-defined auxiliary sends, which are mostly defined and controlled from the game, dynamically.

The following illustration shows a sound routed to an audio bus with 2 additional auxiliary sends. Each auxiliary send has a volume attenuation and is routed to an auxiliary bus.



To add a user-defined auxiliary send

1. Inspect the Properties of an object by double-clicking the object in the Project Explorer

2. Drag & Drop an auxiliary bus from the Project Explorer to the user-defined auxiliary sends list, OR

Click the [...] button to select an auxiliary bus from the Project Explorer selector.

3. Set the send volume to the selected auxiliary bus

Using the User-Defined Auxiliary Sends to control game environments

Auxiliary sends can be used to control environments in the game for simple scenarios. An environment is often defined by the reverberation and the early reflections in an enclosed space. For every sound emitters, the game can control the following elements:

- **Send volume:** which could correspond to the wet portion, or the reflected sounds.
- **Output bus volume:** which could correspond to the dry portion, or direct sound.
- **Output bus low-pass filter:** which could correspond to how much the sound is obstructed or occluded, affecting the frequency response of the direct sound or dry portion.

Attaching a game parameter to the first two values allow you to control the amount of dry and wet signal, per game object. The game would then calculate the distance separating the listener and the sound emitter (game object) and assign that value to the game parameter. In the Game Parameter assignation, the RTPC curves will define how the wet and dry portions of the signal evolve over the distance.

Attaching a game parameter to the Output bus low pass filter would allow you to control how much the sound is occluded or obstructed, from game values.

Using the Game-Defined Auxiliary Sends

Up to 4 game-defined auxiliary sends can be set additionally to the 4 user-defined auxiliary sends. Game defined auxiliary sends are mostly controlled by the game using the following Wwise SDK functions:

- `AK::SoundEngine::SetGameObjectAuxSendValues()`

Use this function to define the send volume to an auxiliary bus for a specific game object. This can be often referred as the wet volume.

- `AK::SoundEngine::SetGameObjectOutputBusVolume()`

Use this function to define the output bus volume. This can be often referred as the dry volume.

However, to enable the game-defined auxiliary send functionality on an object, the option *Use game-defined auxiliary sends* must be activated.

To enable the game-defined auxiliary sends

1. Inspect the object by double-clicking it in the *Project Explorer*
2. In the *Property Editor*:

Go to the *General Settings* tab

3. In the group box *Game-defined Auxiliary Sends*:

Click the *Override parent* if possible

4. Click *Enable game-defined auxiliary sends*

By enabling game-defined auxiliary sends for a specific object, you can control which object is affected by the game-defined sends. In a scenario where game-defined sends are used to control environments, you would be able to control which objects are affected by the environments, and which one are not affected.

When the game-defined auxiliary sends are active, you can make adjustments to the game sends volume values, directly in the Wwise, by changing the game-defined volume, which is additive to the game values.

Using the Game-Defined Auxiliary Sends in combination to the attenuations

Game-defined auxiliary sends can be used in combination to the attenuations, defined in the positioning properties of the object. The attenuation settings allow to control the following game-defined properties from the distance separating the listener to the game object:

- **Game-defined send volume:** allows you to control how much wet signal is attenuated from the distance.
- **Output bus volume:** allows you to control how much dry signal is attenuated from the distance.

Using Loudness Normalization or Make-up gain to Adjust Volume

You may use the make-up gain and automatic source normalization to adjust the volume of each of your audio sources. These volume controls, as opposed to other volume controls in Wwise, are transparent from all logical behavior based on volumes. For example, they have no effect on whether a voice is considered below threshold (virtual), and they are ignored when evaluating HDR attenuation. They don't appear in the Voice Monitor either.

The total value of these source-specific gains is displayed in the column **Normalization / Make-Up Gain** in the **Voices** tab of the **Wwise Profiler**.

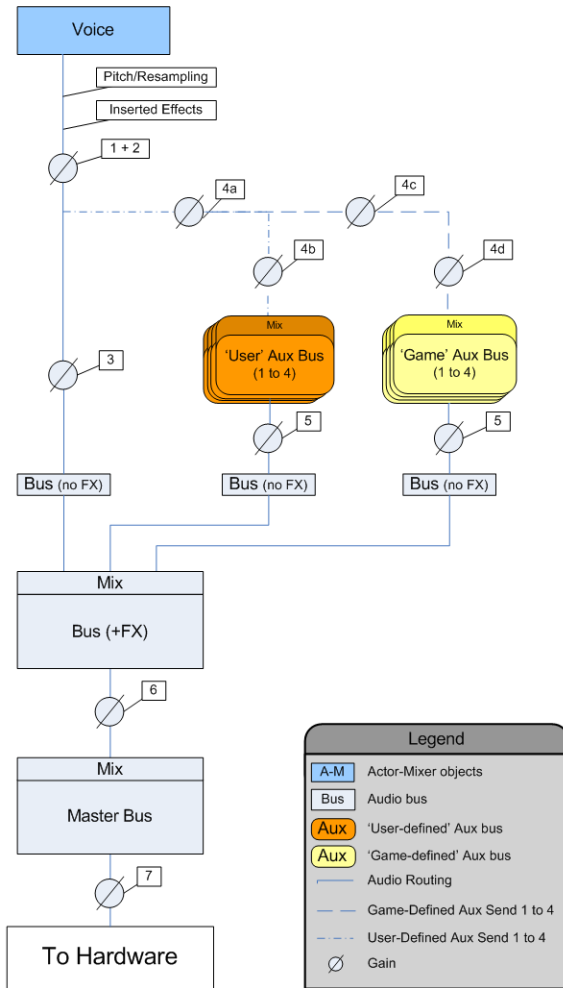
Source Normalization

The loudness data collected in the analysis phase of your original files may be used by Wwise to automatically normalize your assets at run-time based on their estimated loudness. Loudness computation loosely follows the ITU-R BS 1770 recommendation for loudness measurement: K-weighting filtering and absolute and relative gating at -70 dB and -10 dB respectively, on 400 ms windows with 75% overlap.

Source normalization is non-destructive: the sound's analyzed loudness value is stored separately by Wwise, and a proper normalization gain is applied at run-time. This gain is a function of the loudness: $\text{gain} = -\text{loudness} - 23$ [dB]. For example, if a sound's analyzed loudness is -37 dB and source normalization is enabled, Wwise will apply a normalization gain of +14 dB (+37 - 23) at run-time. If a sound's analyzed loudness is -12 dB, the normalization gain will be -11 dB (+12 - 23). In other words, soft sounds are boosted and loud sounds are attenuated. If you are unhappy with the normalization gain that is calculated based on loudness, you may tweak it further using the make-up gain.

Understanding the Voice Pipeline

The following table and diagram shows how voices are being processed, how they are being routed and where the different volumes and effects are being applied.



#	Description
1	<p>Volumes apply to all channels</p> <ul style="list-style-type: none"> • Voice Volume* (<i>Voice Volume</i> parameters on busses and for auto-ducking are applied at this level) • Normalization and Makeup Gain • HDR attenuation • Occlusion <ul style="list-style-type: none"> • Project Setting "Occlusion Volume curve" • API: SetObjectObstructionAndOcclusion() • API: SetAttenuationScalingFactor() • API: SetListenerScalingFactor() <p>LPF property and Occlusion LPF curve (Project Settings)</p>
2	<p>Independent volumes per channel</p> <ul style="list-style-type: none"> • Positioning <ul style="list-style-type: none"> • 2D panning and 3D positioning • Center% • Distance Attenuation Curve "Spread" • API: SetListenerPosition() • API: SetPosition()

#	Description
	<ul style="list-style-type: none"> • API: SetMultiplePositions() • API: SetListenerSpatialization()
3	<p>"Dry Path"</p> <ul style="list-style-type: none"> • Output Bus Volume (+ RTPC) • Distance Attenuation Curve "Output Bus Volume" • API: SetGameObjectOutputBusVolume() • API: AkSpeakerVolumeMatrixCallbackInfo • Obstruction <ul style="list-style-type: none"> • Project Setting "Obstruction Volume curve" • API: SetObjectObstructionAndOcclusion() • Parent Bus without effects inserted: Volume* <p>Obstruction LPF curve (Project Settings)</p>
4	<p>"Wet Path"</p> <ul style="list-style-type: none"> • 4a - <ul style="list-style-type: none"> • Distance Attenuation Curve "Aux Send Volumes" • API: AkSpeakerVolumeMatrixCallbackInfo • 4b - User-Defined Auxiliary Sends Volumes (+ RTPC) • 4c - Game-Defined Auxiliary Sends Volume (+ RTPC) • 4d - API: SetGameObjectAuxSendValues()
5	<ul style="list-style-type: none"> • Bus Volume** • Positioning <ul style="list-style-type: none"> • 2D panning • Center % • API: AkSpeakerVolumeMatrixCallbackInfo • Parent Bus without effects inserted: Bus Volume**
6	<ul style="list-style-type: none"> • Bus Volume** • Positioning <ul style="list-style-type: none"> • 2D panning • Center % • Parent Bus without effects inserted: Bus Volume**
7	<ul style="list-style-type: none"> • Bus Volume**

* Voice Volume = Slider + RTPC + State + Set Voice Volume action

** Bus Volume = Slider + RTPC + State + Set Bus Volume action

Understanding Secondary Outputs

The term "Secondary Output" refers to any of the console's audio outputs that is not the main TV or main speakers output. For these outputs, a separate audio mix must be done depending on the context. The most common secondary output is the speaker on the game controllers (Wii, Wii U, and PS4). Some consoles can have other independent outputs as well (chat, background music, headphones, and so on). For the rest of this section, the discussion will be around the game controller speakers, but the information can apply to all other types of outputs.

To output something on a secondary output, sounds need to be routed to the Master Secondary Bus hierarchy using one of the two following approaches:

- By setting the Output Bus property of a sound directly to any bus in the Secondary Bus hierarchy. This works the same way as any other sound routing. This is the preferred method for sounds that are normally tied to only one secondary output instance. For example, player-initiated gun shots, tennis racket whack, PDA sounds, gameplay feedback, etc.
- Routing a sound through any bus in the Master Audio Bus hierarchy and adding a user or game send to an auxiliary bus inside the Secondary Bus hierarchy. This is the preferred method if the same sound is going to be heard in multiple outputs and/or the TV at the same time. For example: spy camera, announcements, etc.

It is important to know that although there is only one Master Secondary Bus in the project, the bus structure will be duplicated in the game, one copy of the structure for each secondary output (game controllers, companion devices, etc). Therefore, not all the sounds routed into the Master Secondary Bus hierarchy will be mixed together. Effective routing into which copy of the structure will depend on the Listeners and Game Object associations set up by the programmer. This is illustrated in the following examples.

[Secondary Output Examples](#)

Secondary Output Examples

All the examples will be using the bus structure shown below for the Secondary Bus hierarchy. The Main Audio Bus hierarchy is omitted. This structure will be duplicated for each of the outputs so the final mixes are independent. Note that only the required busses will be instantiated.

Figure 29.1. Master Secondary Bus structure

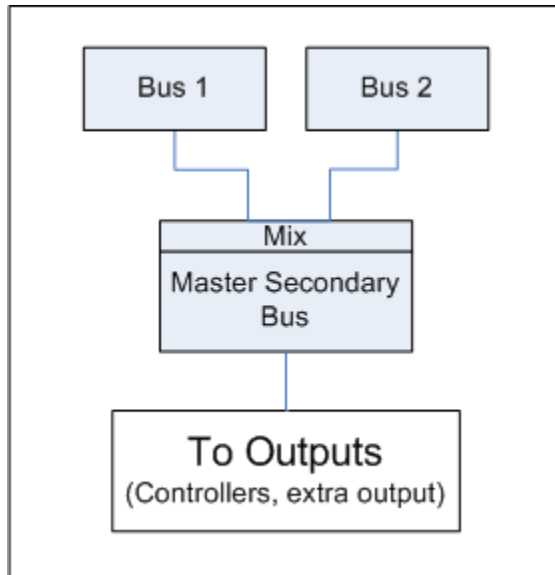
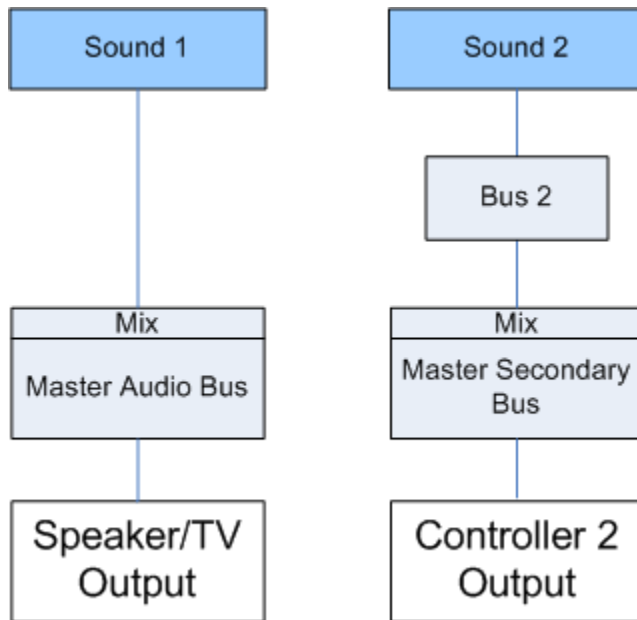


Figure 29.2. Simple example of a sound going to the main output



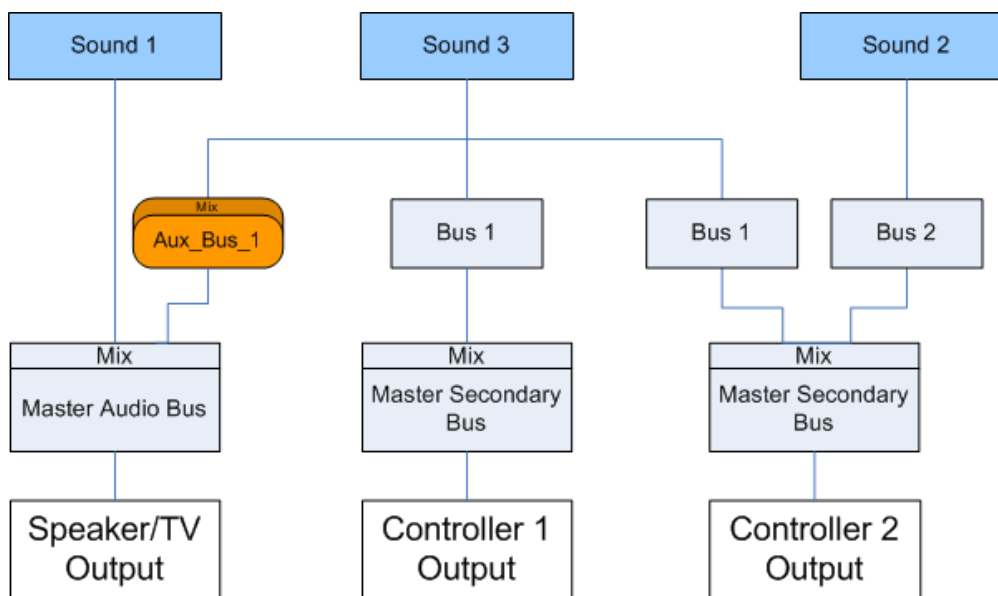
This is a simple example of one sound going to the main output. In a regular game, many of those sounds will be routed to many sub busses. But for brevity, lets assume there is only one sound playing on the TV output.

Figure 29.3. Simple example of a sound going to the second player's controller output



Let's play another sound, this time on the second player's game controller. This could be a menu sound for example, if that player was in the menu, but not the others. In this case, the sound is mixed normally into its output bus and independently of the TV mix. The fact that this sound only play on the player's 2 controller is setup through the game with Listeners and Game Objects association. In this case, the programmer would play the sound on a Game Object that is listened only by the Controller 2's Listener.

Figure 29.4. Sound going to player 1, player 2 and TV.



This example illustrates two points: a sound can be routed to multiple secondary outputs and also to the TV mix. In this example, this could be a prerecorded radio call for help from player 1, which would be heard by player 2 because they are on the same team and also on the TV because the TV point of view is close to the player 2 emitting the sound. Note that this sound's Output Bus is set to Bus 1. This bus is duplicated for player 1 and 2, as is the Master Secondary Bus. Clearly, the player 2's mix has to include Sound 2 (which he is the only one to hear) as well as Sound 1. Both copies of the Master Secondary Bus will have the same effects applied (if any), but won't do so on the same audio signal.

You can also notice the Send going from Sound 3 to the TV mix. All types of Sends can be used to send to another output. In this case, since the call may be heard by the "camera" too (depending on distance and attenuation), so it needs to be mixed into the main TV mix and will "enter" that mix from the defined auxiliary bus.

Understanding HDR

High dynamic range audio (HDR audio) is a technique to design a mix using level values spanning across a very high dynamic range as occurs in nature. HDR is also a run-time system that dynamically maps this wide range of levels to a range that is more suited to your sound system's digital output.

In the real world, the audible dynamic range that spans from the threshold of human hearing to the loudest possible sound in air is several times wider than the dynamic range offered by speakers at game play levels. The role of an

HDR system is to collapse or "compress" the whole real life dynamic range, approximately 190 dB, into 96 dB (the dynamic range available for a digital device), and even less in practice due to floor noise levels.

In HDR photography, local tone mapping is applied independently to various regions of an image to enhance the contrast within each region. HDR audio works in the same way; it performs sound level mapping instead of tone mapping, and does so locally in time. Thus, at any given moment, the system automatically adapts the mapping based on the levels of sounds that constitute the audio scene.

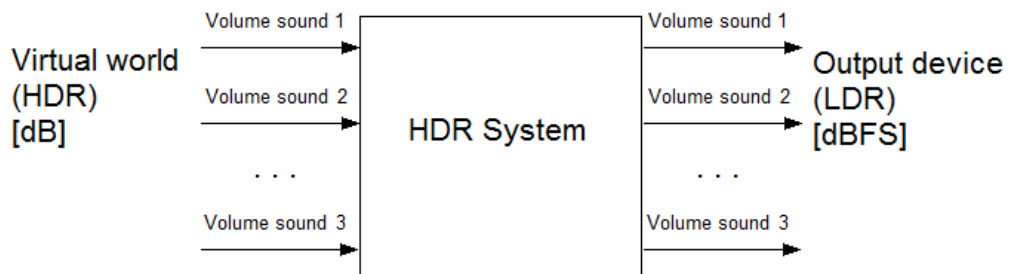
HDR Glossary:

Term	Definition
Decibel (dB)	A logarithmic measure of the level of a sound compared to the level of another sound or an arbitrary reference value. One decibel equals $20 * \log_{10}(A/AR)$ when dealing with amplitude. A difference of +20 dB means that a sound's amplitude is 10 times greater than the reference.
Decibel full scale (dBFS)	A logarithmic measure of the amplitude of a signal compared with the maximum that a device can handle before clipping occurs. A value of 0 dBFS is thus the loudest sound that can be generated by the digital audio output. 16-bit digital audio output devices range from 0 dBFS down to -96 dBFS. The level of the audio signal coming out of the Master Audio Bus in Wwise should therefore lie between these values.

HDR examples

In HDR audio, you can assign volume values to sounds of the game's virtual world that span over a much larger dynamic range than the standard 96 dB of 16-bit output devices, much like they would in the real world. It is the task of the HDR system to translate these values into dBFS, as illustrated in the figure below.

Figure 29.5. HDR system inputs and outputs



The inputs are the sound levels of the virtual world, expressed in decibels (dB) relative to an arbitrary reference. The values can be chosen arbitrarily

high or low, and thus have a high dynamic range. The outputs are the levels of the corresponding sounds in dBFS. The range of these values depends on the output device, which typically has lower dynamic range than the input.

In its simplest form, the HDR system operates as follows: at each time slice, the system selects the sound assigned the highest volume in the virtual world, automatically maps it to the output value of 0 dBFS, and then maps all other sounds proportionally.

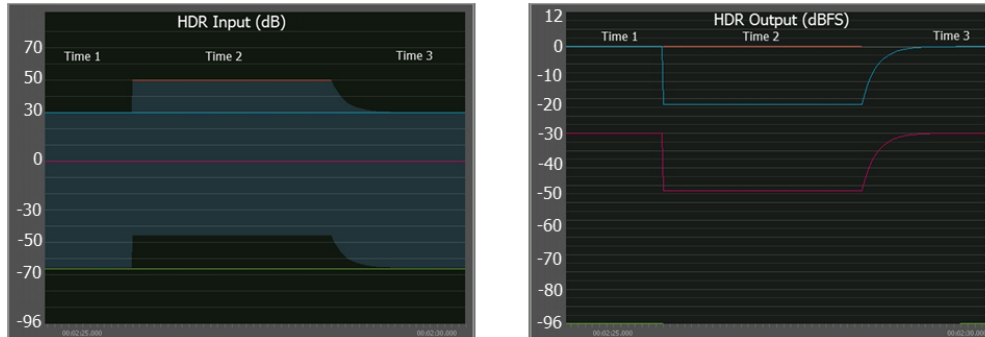
Let's use an example to illustrate this. Suppose that at a given moment ("time 1"), sound "blue" plays at +30 dB in the virtual world, as illustrated on the left side of [Figure 29.6, "HDR window"](#). The reference (0 dB) is arbitrary. Because "blue" is the loudest sound at time 1, it plays at 0 dBFS at the output of the HDR system. Another sound, "purple", plays at 0 dB in the virtual world, that is, 30 dB below sound "blue". Thus, it comes out at -30 dBFS at the output of the HDR system. A third sound, "green", plays at -66 dB in the virtual world, which results in -96 dBFS at the output of the HDR system. Since the output of the system is constrained to a dynamic range of 96 dB, the level of "green" corresponds to the lower bound of all audible sounds. At time 1, any sound that is softer than "green" is inaudible.

In the previous example, the range [-66, +30] dB in the virtual world (input side) is referred to as the **HDR window**. It is represented by the blue region on the left side of [Figure 29.6, "HDR window"](#). The HDR window has a fixed width, determined by the dynamic range of the output. For a 16-bit device, it is equal to 96 dB at most, but in practice it is usually smaller. At time 1, the sound at +30 dB is the loudest in the virtual world, and any sound below -66 dB is inaudible because it is below the HDR window.

Suppose that later, at time 2, another sound, "orange", starts playing at +50 dB in the virtual world. To accommodate this new louder sound, the HDR system slides the window up by 20 dB, so that its bounds are now [-44, +50] dB on the input side. All sounds are then mapped to the new values. The sound at +50 dB plays at 0 dBFS, the sound at +30 dB now plays at -20 dBFS, and the sound at -66 dB is now below the window and is therefore completely inaudible. When the orange sound stops playing at time 3, the window gently slides back to where it was, and other sounds take their former volume.

Figure 29.6. HDR window

	Time 1		Time 2		Time 3	
	Input	Output	Input	Output	Input	Output
Orange	-	-	50	0	-	-
Blue	30	0	30	-20	30	0
Purple	0	-30	0	-50	0	-30
Green	-66	-96	-66	-∞	-66	-96



On the left, sound levels are represented at the input of the system, in decibels (dB) with arbitrary reference. The HDR window is represented by the blue region. At time 1, the top of the window is aligned with the loudest playing sound (blue). At time 2, another sound (orange) starts playing at +50 dB, and the window slides up immediately to accommodate it. The sound at -66 dB (green) is then clearly below the window, and thus inaudible. The resulting levels in dBFS at the output of the system are shown on the right. When the HDR window slides up by +20 dB because of the orange sound, the volume of other sounds drops by -20 dB. During this time, the green sound is completely excluded from the output. Notice that the orange sound plays at the same output level as the blue sound during time 1. When the orange sound stops playing, the window gently slides down to its former level, and the volume of other sounds increases accordingly.

The HDR system works like a dynamic range limiter/compressor. It affects your mix by making soft sounds inaudible when loud sounds play, and making them audible again when playing alone. The relative levels of sounds between one another in the HDR world are preserved, creating the illusion of a greater dynamic range, while in fact they are compressed within the output device's lower dynamic range. Furthermore, thanks to the system's automatic volume ducking when louder sounds play, your mix will be cleaner and have better focus. The next figure illustrates this principle.

Figure 29.7. HDR overview, as seen on the input side of the system



The window slides up only when louder sounds play. When the window slides up on the input side, the sound volume drops on the output side. What was formerly audible, such as the sound of leaves in a tree, can become completely inaudible when a gunshot is played. The actual volume of sounds at the output of the system depends on the distance between them and the top of the window, at any given moment. Here, the gunshot and explosion would come out of the system at the same level if played alone, but because the explosion is considered louder than the gunshot and effectively ducks its volume, listeners are left with the impression that it is indeed louder.

Using HDR

In Wwise, users need to select a bus to act as a converter between HDR levels and full (device) scale. The input levels that this HDR bus consumes are the logical levels that you set in Wwise. Thus, sounds routed to an HDR bus can have their volume set far beyond 0 dB. The only thing that counts is their position within the HDR window, which is placed dynamically by the HDR bus according to what is playing. The HDR bus therefore acts as a logical limiter/compressor by exposing controls that are similar to those of an audio limiter/compressor. It has ballistics (infinitesimal attack, user-defined release) to control how the HDR window slides in time. It also has a threshold, which can be seen as the lowest possible position of the HDR window.



Usage of dB SPL

In prior literature, HDR audio systems often express input side volume levels in terms of Sound Pressure Levels units (dB SPL). dB SPL are a measure in decibels whose reference (0 dB SPL) corresponds to the threshold of human hearing. The notion of SPL does not exist in Wwise because it adds unnecessary complexity, pollutes the interface, and does not make the system more usable. Instead, the input side reference is left arbitrary, and it is up to you to define it. If you wish to work in dB SPL, you can set the volume of sound structures to positive dB SPL values directly. On the other hand, the default range of volume sliders in Wwise goes up to +12 dB only, so it might be more practical to choose another reference, and make the necessary subtraction to find the corresponding relative dB level from the desired SPL value. For example, you may decide that 100 dB SPL is your reference at 0 dB. Then a sound at 80 dB SPL needs to have its volume slider set to -20 dB, a sound at 130 dB SPL needs to have its volume slider set to +30 dB, and so on. You also need to set the HDR bus threshold accordingly.

- [Enabling HDR in your Project](#)
- [Routing Sounds to an HDR Bus](#)
- [Monitoring the Window](#)
- [Setting the Dynamics of HDR](#)

Enabling HDR in your Project

To enable HDR in your project, you simply need to define which bus or busses will act as HDR subsystems.

To enable HDR on a bus:

1. Inspect a bus object.
2. Click the **HDR** tab of the **Property Editor**.
3. Click **Enable HDR**.

The volume of all sound structures routed to the HDR bus, or to one of its descendants, is affected by the HDR bus. Sounds that are not routed to the HDR bus are not, but can coexist with sounds routed to the HDR bus within your sound design. You can also have more than one HDR bus in a project, as long as they are completely independent: an HDR bus may not have another HDR bus as an ascendant or descendant.

As mentioned previously, sounds at the top of the HDR window will come out at 0 dBFS. Use the bus volume slider of the HDR bus to scale down the output

of the HDR system before mixing it with other non-HDR sections of your project.

Routing Sounds to an HDR Bus

To have sound objects in the Actor-Mixer Hierarchy or music hierarchy use the HDR system, you need to route these objects to the HDR bus.

To route a sound to an HDR bus:

1. Inspect the sound object to show its content.
2. In the **Property Editor**, go to the **General Settings** tab.
3. Select the HDR bus as the **Output bus**.

Monitoring the Window

The Voice Monitor view displays the volume of voices and their envelope (if available).

To understand and debug the HDR system:

1. Open the **Voice Monitor** view.
2. Drag your HDR bus into the view's context.
3. Set the Mode to either Bus input or Bus output. Bus input mode displays voice levels in decibels as seen by the HDR bus, before the HDR compression, while Bus output mode displays them at the output of the HDR bus, after HDR compression and the bus output gain. Most figures in this document are screenshots of the "two sides" of the Voice Monitor view.

Setting the Dynamics of HDR

Each HDR bus maintains an HDR window whose position and width is defined by the level of the loudest sound at any moment, and the project's volume threshold respectively. The behavior of the HDR window can be edited in the HDR tab of each HDR bus. Since the HDR bus acts like an audio limiter/compressor, its controls are similar.

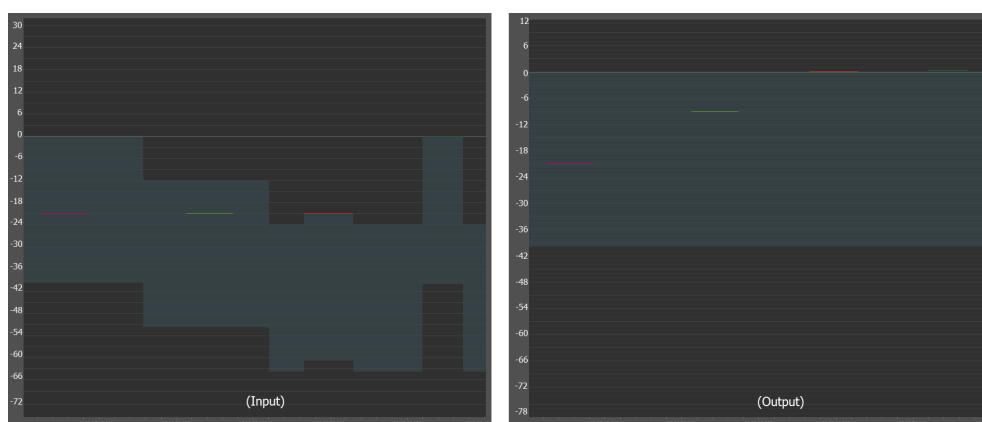
- [Working with the HDR Threshold](#)
- [Working with HDR Ballistics](#)
- [Working with the HDR Ratio](#)

Working with the HDR Threshold

Like the threshold of a typical audio limiter/compressor, the HDR threshold defines the minimum input level above which the HDR window may slide

(refer to the yellow line in [Figure 29.7, “HDR overview, as seen on the input side of the system”](#)), or in other words, the minimum level above which the compressor "kicks in". When only soft sounds are playing, it directly impacts their level at the output: the farther away they are from the window top, the lower their output level is. When sounds play above this threshold, then softer sounds are ducked down automatically. It is important to understand that with an infinite compression ratio (we will discuss about the ratio later), any two sounds whose input level is above the HDR threshold will play back at the same output level when played alone, regardless of their input level as shown in [Figure 29.8, “Effect of HDR threshold”](#).

Figure 29.8. Effect of HDR threshold

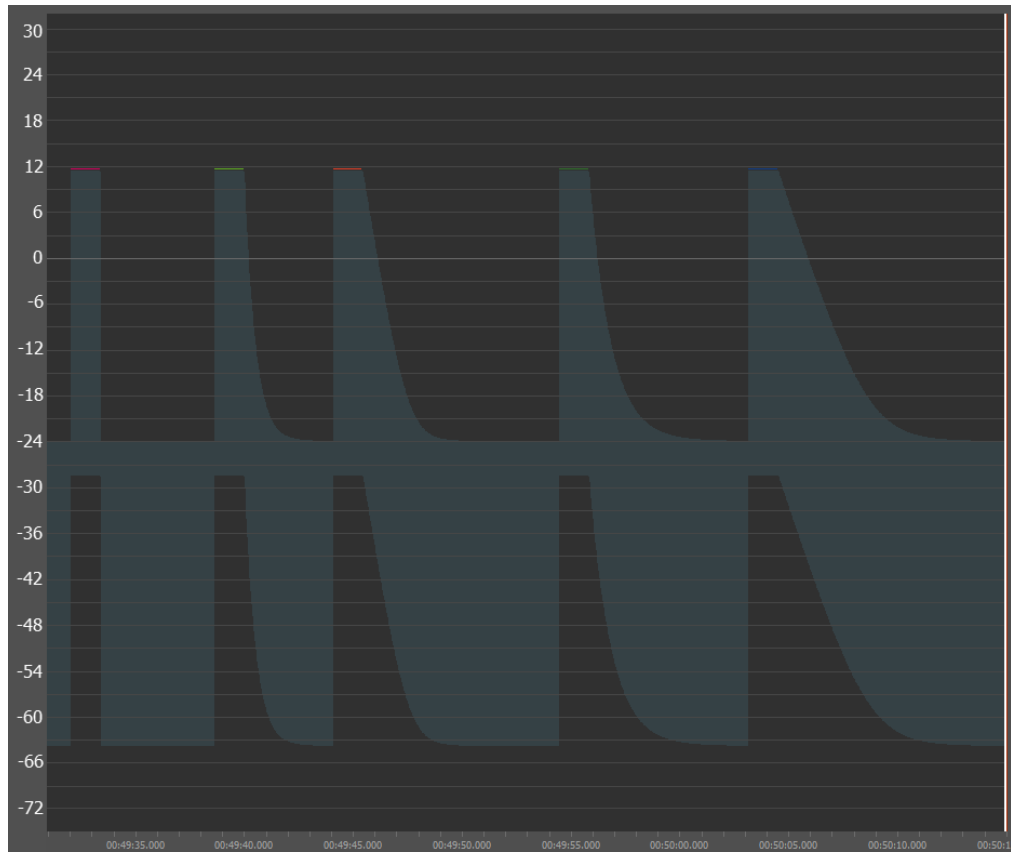


On the figure above, the left side shows input; the right side shows output. The first time, the sound is set at -21 dB with a threshold of 0 dB, and comes out at -21 dBFS. The second time, the threshold is at -12 dB and the sound comes out at -9 dBFS. The third time, the sound is above the threshold of -24 dB, and thus drives the window and comes out at 0 dBFS. The sound is replayed at an input level of 0 dB, even higher above threshold, and still comes out at 0 dBFS

Working with HDR Ballistics

Standard audio compressors implement ballistics (attack/release) that define how compression behaves in time. Likewise, the HDR bus has user controls for the release time and shape, which let you define how the HDR window releases to a lower value. [Figure 29.9, “HDR release: 0, 0.5 exponential, 0.5 linear, 1 exponential, 1 linear”](#) shows various release times and modes. Adjust it to minimize artifacts such as pumping. Note however that there is no control for attack time, because the attack of an HDR system has to be instantaneous, like an audio limiter.

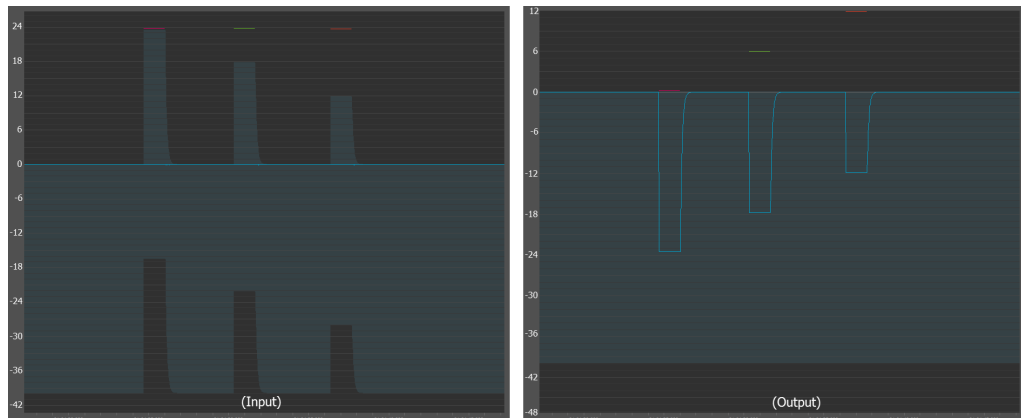
Figure 29.9. HDR release: 0, 0.5 exponential, 0.5 linear, 1 exponential, 1 linear



Working with the HDR Ratio

So far, the HDR system has been presented like an audio limiter. With the HDR window top following exactly the level of the loudest sound, every sound above the HDR threshold is played back at the same output level when played alone, as was displayed in [Figure 29.8, “Effect of HDR threshold”](#). As previously described, while HDR increases the perceived dynamic range between sounds relative to each other, it makes the "absolute" dynamic range smaller. It is possible to gain some of this "absolute" dynamic range back by transforming the HDR limiter into an HDR compressor. You do this by using a finite, smaller compression ratio (it is infinite by default). The compression ratio defines how closely the HDR window follows the loudest sound, and is a function of the difference between the sound's level and the HDR threshold. [Figure 29.10, “Effect of HDR ratio”](#) illustrates the effect of the ratio. Note that with non-infinite compression ratios, loud sounds may overshoot the HDR window, and thus come out of the HDR system above 0 dBFS. Give yourself enough headroom by lowering the volume of the HDR bus and/or other busses downstream in the signal path.

Figure 29.10. Effect of HDR ratio



The sound has an input level of +24 dB (left panel), and played in HDR bus with ratio 100:1, 4:1, 2:1. The corresponding output levels of 0 dBFS, + 6 dBFS and +12 dBFS are displayed on the right side. Another sound has an input level of 0 dB and is ducked down by -24 dB, -18 dB and -12 dB. At 2:1, only half of the energy above threshold is used to drive the HDR window up, hence ducking the other sounds by -12 dB. Notice that the level difference of 24 dB is preserved all along. Also, when using smaller ratios, sounds may stand above the window top, hence above 0 dBFS, so you should reserve sufficient headroom by decreasing the volume of the bus or its parents.

Working with Amplitude Envelopes

Since the HDR system works with logical volume values set in Wwise, it is unaware of the actual amplitude of the input sounds, and therefore sees them as black boxes with constant volume throughout the duration of the sound. Most of the time, the amplitude of sounds vary. Imagine an impact sound with a transient and a decaying part. If this sound was loud enough to fix the position of the HDR window, this position will remain unnaturally constant during the whole duration of the sound. See the resulting effect on the window in [Figure 29.11, “HDR window with a decaying impact sound without envelope”](#).

In Wwise, it is possible to let the HDR system peek into the black box by enabling envelope tracking. In the HDR tab of the desired sound's properties, select the **Enable Envelope** check box: the amplitude envelope of the audio file is then analyzed and is attached to the sound's metadata. At run-time, the HDR system uses this data to move the window appropriately, as can be seen in [Figure 29.12, “HDR window with a decaying impact sound with envelope, playing above a softer, steady background sound”](#).

For the following figure, the input is on the left, the output on the right and the corresponding output wave is below.

Figure 29.11. HDR window with a decaying impact sound without envelope

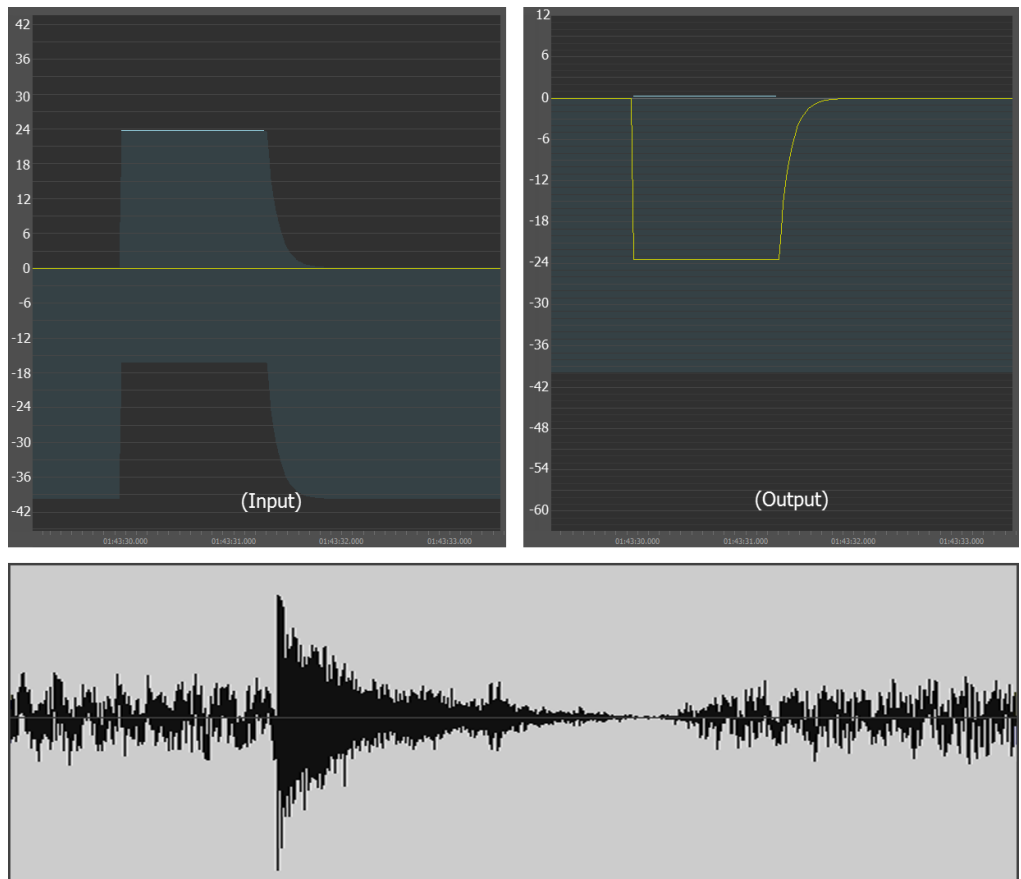
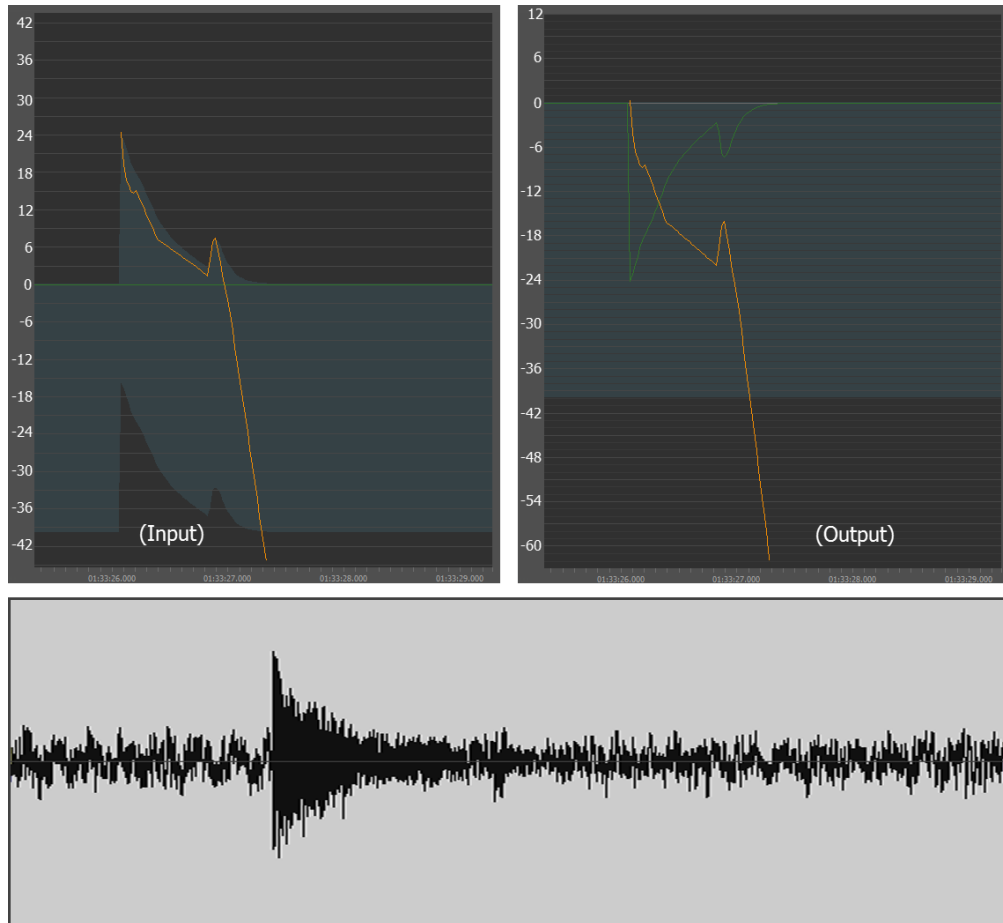


Figure 29.12. HDR window with a decaying impact sound with envelope, playing above a softer, steady background sound



Notice the lull after the impact sound in the first figure above, caused by the HDR system interpreting it as being constant.

Envelope Sensitivity and Manual Editing

Once you enable the envelope, you can preview the result by opening the source editor (next figure). If you are unhappy with the precision and/or number of points, you can change the value of the sensitivity slider in the envelope tracking group box (see [Figure 29.14, “HDR tab in Property Editor”](#)). You can work with a group of sounds by enabling the envelope and setting the sensitivity on a higher-level sound structure, an actor-mixer for example. Note that the effect of the sensitivity is dependent on the original audio data, and may therefore be different for different variations of similar sounds. You can also edit the envelope manually for each sound; set the sensitivity control so that you obtain a curve that is close to the desired result, and move, remove or add new points.

Figure 29.13. Source editor in RMS mode, with envelope display

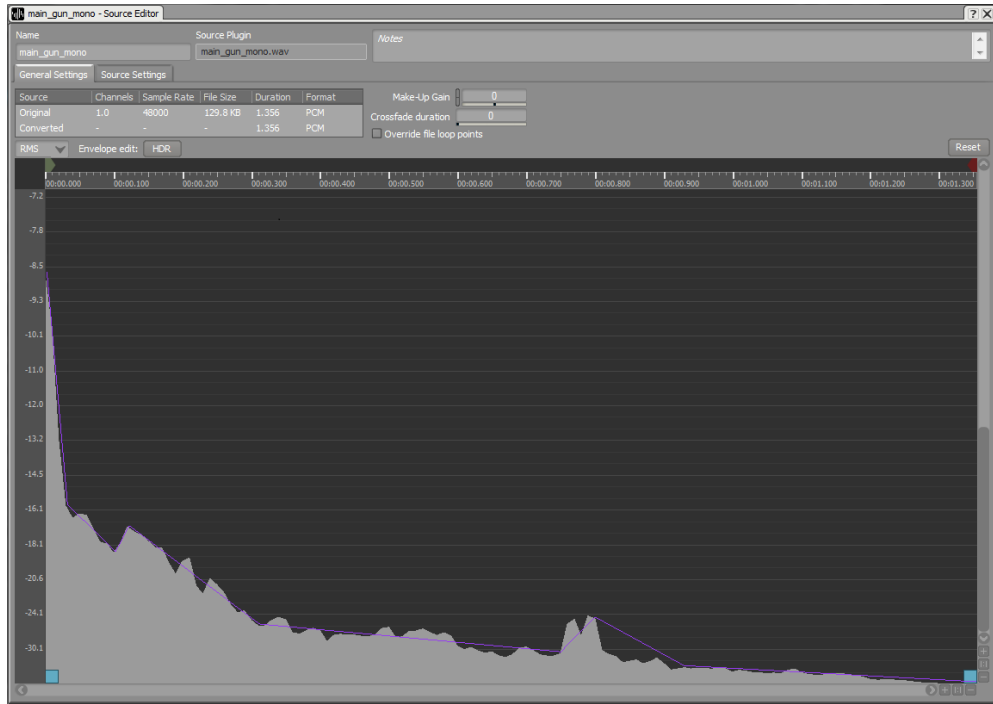
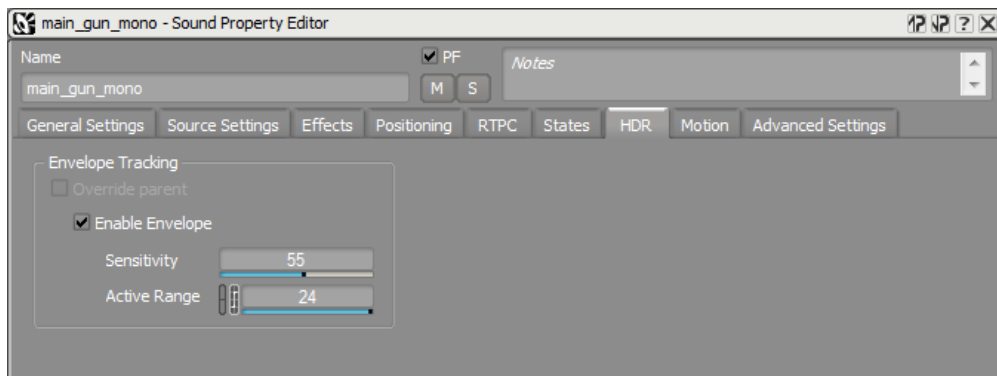


Figure 29.14. HDR tab in Property Editor



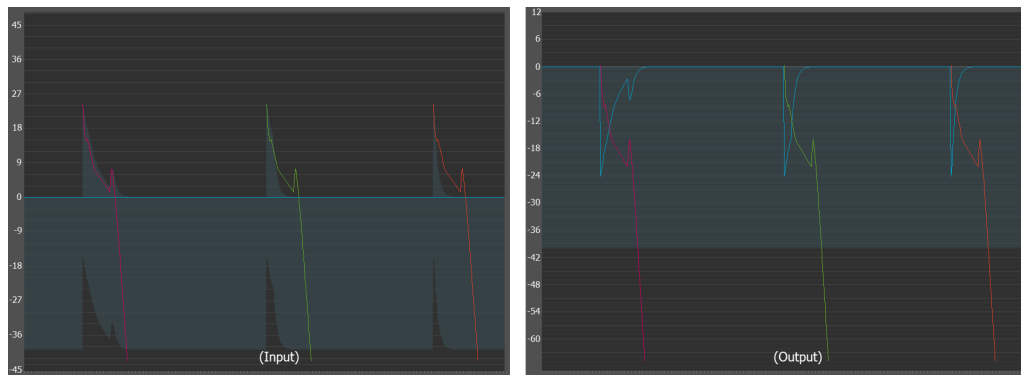
Region of Interest: Active Range

It has been shown that in typical cases, sounds that have large level differences such as an impact sound that ranges from 0 dB to -96 dB should not drive the HDR window for their whole duration, but should instead do so only during a limited period of time. The rationale for this comes from the following paradox: HDR acts as a priority system based on loudness, but the softer parts of a loud sound should not be given as much importance. For example, the decaying tail of a grenade sound has very low importance compared to the transient, and thus should not cover up the impact of a shotgun, even though the former is

technically louder, on an absolute scale. In this case, the region of interest of the grenade sound should be limited to its first impact. It moves the HDR window up and therefore ducks everything else. On the other hand, once in the tail and playing say, 12 dB below the peak value, it is usually not desired to continue ducking the other sounds, even if the grenade's volume minus 12 dB is still above the level of all the other sounds in the audio scene. Restricting the HDR window control to a region of interest can be regarded as a way of gracefully blending together various "tonal regions" to continue with the HDR imaging analogy.

In Wwise, the way to define the region of interest is to declare a range in decibels from the peak of the sound, the active range. When the sound's envelope drops below the active range, the sound is considered outside of its region of interest, and is not considered by the HDR system as being able to drive the HDR window. [Figure 29.15, "Active range"](#) illustrates this with three similar sounds having different active ranges.

Figure 29.15. Active range



The same decaying sound is played three times with active range set to 96 dB, 12 dB and 6 dB respectively, above a steady background sound. When the sound drops by that amount of decibels below its peak, the window stops following it and instead releases back to idle. The movement of the window distinctly affects the level of the background sound. On the other hand, the window or active range has no impact on the decaying sound itself.

More about HDR

It is important to realize that the envelope of a sound only has an effect on other sounds, and never on itself. This last characteristic is what mostly distinguishes the HDR system from an audio compressor. At any given time, the loudest sound is scaled by its peak value, but is unaffected by its own envelope, which contributes to making the HDR system sound transparent. In consequence, level differences between input sounds are not strictly preserved when envelopes

are used. This has the benefit of blending sounds of varying "local loudness" together in the audio scene, as was discussed in the previous section. But you may find that the audio scene loses a bit of its realism when the HDR window moves wildly. In HDR imaging, the same situation arises when using a large amount of smaller tonal regions. Lots of examples of unrealistic HDR photos can be found on the web. We thus recommend that you proceed carefully in setting levels above the HDR threshold: do not just pluck in values blindly, but instead carefully set the level and design the envelope of your loudest sounds in order to make space for them in the mix.

Avoid enabling envelopes for nothing, as they require extra memory. Envelopes are useless for soft sounds that never go above the HDR threshold, so make sure to disable them.

Beware of envelopes that wiggle above and below the active range, as this might cause unwanted behavior. In the following figure, a long explosion sound goes in and out of the active range, provoking rapid changes of the HDR window. In this case, you might want to coarsen the approximation of the envelope and/or edit it manually. You may even want to proceed artistically and design an envelope that does not exactly correspond to the reality, by ducking the audio scene only during the initial impact of the sound, and recovering sooner during the rumbling part.

Simpler envelopes with fewer points result in less erratic amplitude changes of the audio scene, and this is often preferable.

Figure 29.16. Editing envelopes (a)

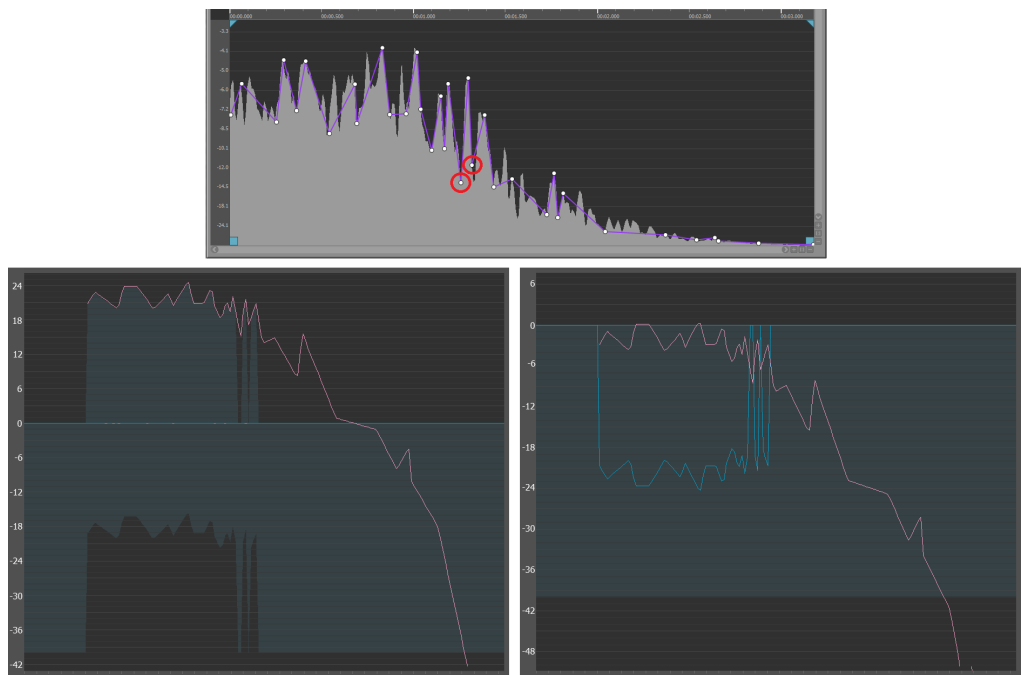


Figure 29.17. Editing envelopes (b)

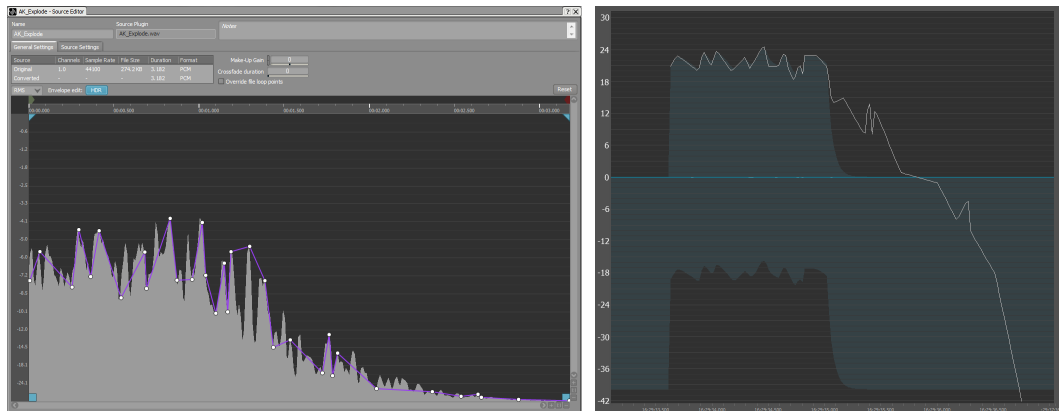
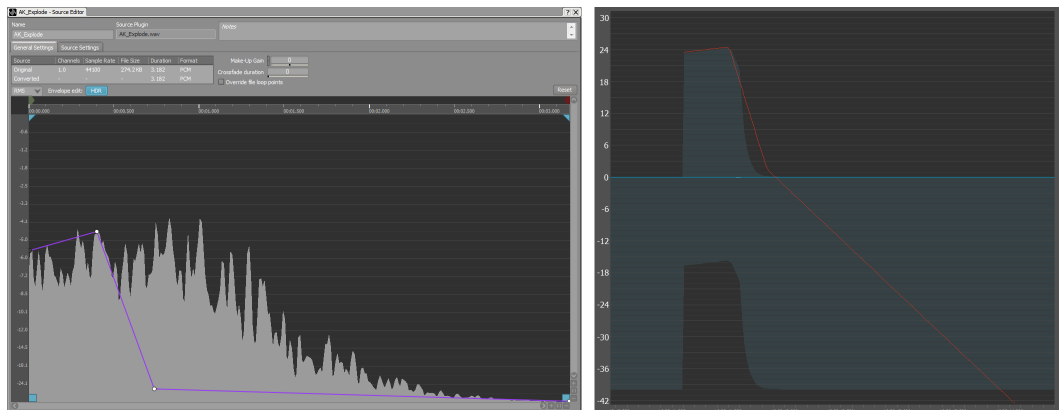


Figure 29.18. Editing envelopes (c)



In (a), the envelope of an explosion sound is displayed in the source editor. The points circled in red are below the active range. During playback, the envelope goes in and out of the region of interest, resulting in rapid movements of the HDR window. The drastic effect on the volume of the background sound is very explicitly illustrated in the resulting output. To fix this problem, the envelope was edited manually in (b) by removing the two offending points. Recall that editing the explosion envelope only affects the HDR window, that is, other sounds, but not the explosion itself. For aesthetic reasons, the designer may also decide to duck other sounds only during the initial impact of the explosion, but let them take their full volume earlier, during the following rumble by editing the envelope somewhat like in (c). Interestingly, in all three examples the explosion is played back identically.

HDR and the Wwise Voice Pipeline

As previously mentioned, the HDR system works with logical volumes set in the Wwise authoring tool, and ignores the actual amplitude of audio data. The levels considered by the HDR system's logic correspond to those at the input

of the HDR bus, as can be seen in the Voice Monitor when inspecting the HDR bus in Input bus mode. These levels depend on the voice volumes, which are the sum of contributions from the actor-mixer hierarchy, control busses of the master-mixer hierarchy, actions, RTPC and distance attenuation, as well as on gains of each mixing bus located upstream of the HDR bus in the signal flow. When multiple signal paths lead to the HDR bus, for example when using auxiliary sends, the path with maximum gain is used by the HDR logic.

At each audio frame, the sound engine begins by computing the volume of all voices as it is seen by the HDR bus. It then executes the HDR system's logic, which returns the global HDR gain/attenuation according to the position of the HDR window, and reapplies this gain into each voice. Once this is done, it processes voices as usual: voices are evaluated against the volume threshold to decide if they should be virtual, data is produced, and volumes are applied.

Make-Up Gain and Source Normalization

The HDR system's logic ignores two volume properties supplied by Wwise: source normalization and make-up gain(s). These volume controls are primarily used to normalize the audio assets independently of their logical volume, as you could have done in your wave editor prior to importing them into Wwise. For example, you can use the localization make-up gain to even out volume differences between two languages, but the HDR system should have the same behavior across all languages. Thus, it is important to have volume controls that are ignored by the HDR logic, and source normalization and make-up gains play this role.

Note that the virtual voices system, as well as the Voice Monitor calculations, does not take into account the make-up gain and the source normalization. However, virtual voices do take the HDR attenuation into account. Thus, voices can become virtual when they are below the HDR window.

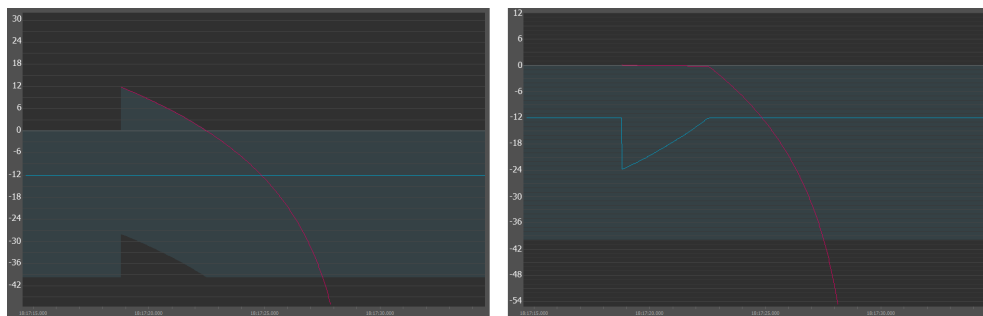
The make-up gain slider in the Source Settings tab of the Property Editor can also be used with HDR for aesthetic purposes. Since it is transparent from the HDR system's point of view, it can be seen as a post-HDR volume (in terms of signal flow), allowing you to change the volume of sounds independently of the HDR window. Normally, any sound playing alone above the HDR threshold comes out of the system at the same level. With the make-up gain, you can effectively make it louder. For example, you might want the player's guns of your first-person shooter to be much louder than the rest, while benefiting from the HDR system's dynamic mixing capabilities. In this case, give them a boost using the make-up gain.

HDR and Distance Attenuation

Distance attenuation curves can be designed as usual. However, be aware that there is a somewhat unexpected behavior caused by the HDR system.

Recall that when you play a sound that is above the HDR threshold and is the loudest sound of the audio scene, it comes out of the system at 0 dBFS. If that sound is played back 50 meters away, but remains above threshold and remains the loudest one in your scene, it will still come out of the system at 0 dBFS. Consequently, you will be left with the impression that the attenuation curve does not work at closer distances. It is not the case. What happens is that the increasing input volume at closer distances is actually used to duck other sounds instead of increasing the output volume. If nothing else is playing, you will not notice this, but you need to keep it in mind when designing attenuation curves for louder sounds.

Figure 29.19. Effect of distance on loud sounds



Here, the listener is steadily stepping away from a loud sound (in red). While the volume seems correct on the input side (left), the output (right) exhibits a plateau when the loud sound remains above the HDR threshold in spite of the increasing distance, giving the false impression that distance attenuation is not working. In fact, at closer distances, the volume offset exceeding the threshold affects other sounds (in blue) instead of the sound in question.

Using the HDR Window as an Input Variable

The HDR window position can be monitored by looking at the window meter in the HDR tab of a bus. You can also assign a game parameter to its value. From there, you can map the game parameter to any property of any object via RTPC. For example, you could use the window position to tweak the maximum number of sound instances.

To attach a game parameter on the HDR window position:

1. Inspect the HDR bus to see its content in the **Property Editor**.
2. Click the **HDR** tab.
3. In section **Window Top Output Game Parameter**, click [...] to browse for a game parameter.
4. Set the range for which the game parameter will be set, in decibel on the HDR bus input.

Creating the Final Mix

After you have set up the bus structure in your project, you can fine-tune and troubleshoot the mix of sounds, music, and/or motion in the game. Wwise has two main tools for mixing your game audio:

- The [Master-Mixer Console](#) - A mixing console that groups a variety of bus properties into one view.
- The [Mixing Desk](#) - A flexible and powerful mixing console that groups a variety of bus and object properties into one view.

Both of these views can be used to create the final audio mix of your game in real time. The Mixing Desk, however, gives you more control because more properties are available for editing, and both objects and/or busses can be included in a mixing session.

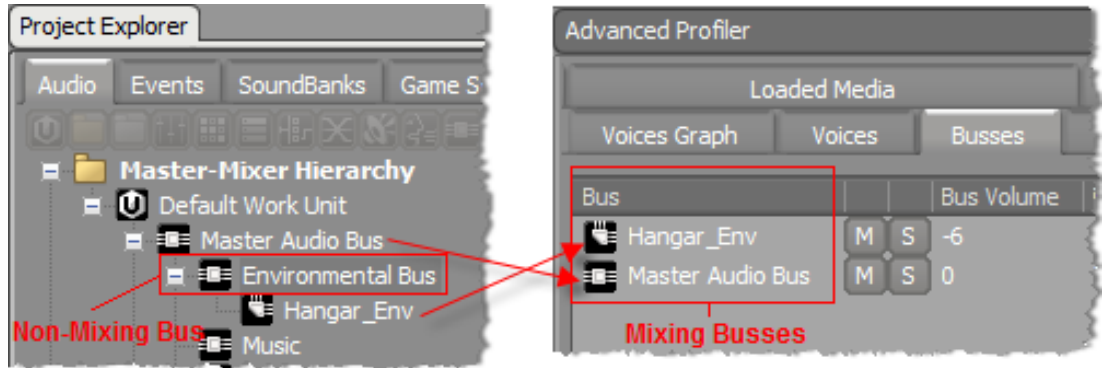
Mixing Versus Non-Mixing Busses

In Wwise, Master-Mixer Hierarchy objects (busses) are handled under the hood as non-mixing busses, unless they need to be converted to mixing busses. Non-mixing busses propagate their bus properties (such as the bus volume) to their inputs because it is a more efficient use of CPU and memory. A non-mixing bus is promoted to a mixing bus when such propagation would not work because:

- It is a Master Audio Bus;
- It is an Auxiliary Bus;
- It is an HDR bus;
- It has at least one inserted effect;
- It is mounted with a mixer plug-in;
- Its channel configuration is not set to **Parent**; or
- It has positioning enabled.

One way to verify the status of the bus is by inspecting the **Voices Graph** tab or the **Busses** tab of the Advanced Profiler while capturing. Only mixing busses appear there; non-mixing busses do not.

As seen below in the screenshot taken from the Wwise Integration project, the Environmental Bus must be non-mixing because it does not show up in the Advanced Profiler's **Busses** tab while its child Hangar_Env Auxiliary Bus does. Incidentally, as a non-mixing bus, the Environmental Bus's volume is included in the Hangar_Env Auxiliary Bus **Bus Volume** column.



Monitoring the Signal Level

Loudness Overview

The ITU-R BS.1770 and EBU R128 standards provide recommendations and specifications for measuring the loudness level and loudness range of a given audio program in loudness units (LU or LUFS). It also provides specifications and recommendations for the True Peak level. Wwise implements these specifications and recommendations in the Loudness Meter view and Meter view.

Measurements

Wwise offers different measurements of the PCM signal:

Measurement	Description
Loudness level	<p>Provides a measurement the perceptual loudness of a digital signal. The loudness level can be calculated using different time window sizes.</p> <p>The Loudness level is available in the Loudness Meter view.</p> <p>Units: LUFS or LU</p>
Loudness range	<p>Provides a statistical analysis of the loudness values over a period of time and a measure of the dynamic range.</p> <p>The Loudness range is available in the Loudness Meter view.</p> <p>Units: LU</p>
True-Peak level	<p>Provides an estimate of the maximum continuous signal level in the analogue world. This value is useful to ensure that the digital to analogue converter or an external codec is not saturated.</p> <p>The True Peak level is available in the Meter View.</p> <p>Units: dBTP</p>
Peak level	<p>Provides a measurement of the maximum or minimum values of a PCM signal for a very short period of time.</p> <p>The peak level is shown at the following locations in Wwise:</p>

Measurement	Description
	<ul style="list-style-type: none"> • Wwise Toolbar • Master-Mixer Console • Mixing Desk • Meter view Units: dBFS
Root mean square (RMS)	The RMS level is measured with the peak meter values and does an average over a very short period of time. The RMS level is available in the Meter View . Units: dBFS

Loudness Units

The EBU R128 and ITU-R BS.1770 define units for measuring loudness. Loudness is not measured in dBFS.

Unit	Description
Loudness Unit (LU)	0 LU is equal to the reference level (by default -23 LUFS). Note: The default value of 23 LUFS can be changed in the Loudness Meter settings.
Loudness Unit Full Scale (LUFS)	0 LUFS/LKFS is equal to 0 dBFS @ 1KHz according to ITU-R BS.1770-3 and EBU R128
LKFS	LKFS is the same as LUFS. ITU-R BS.1770-3 are using the terms LKFS. EBU R128 and using LUFS.

Loudness Measurement Scopes

Loudness can be measured using three different time scopes. In the integrated mode, the loudness level of a game can be determined at a larger scope.

Term	Description
Momentary	Loudness level calculated using a sliding rectangular time window of length 0.4 seconds. The measurement is not gated.
Short-term	Loudness level calculated using a sliding rectangular time window of length 3 seconds. The measurement is not gated.
Integrated	Loudness level calculated using the user capture. The integrated loudness uses gating (all values below -70 LUFS are ignored) and is a long term measurement (an integration over the entire capture time) based on the momentary loudness. The measurement starts when clicking the Capture button, and stops when clicking it again. Clicking the Reset button restarts the capture.

More on Integrated Loudness

EBU R128 recommends a target level of -23 LUFS for integrated loudness. Selecting -23 LUFS specifically for a game as the target loudness level helps

game users obtain a more consistent output level on the output device (television, mobile device etc.) they are using, with other sources. Selecting a specific loudness target level for the whole game also ensures consistent loudness over the different levels or parts of the game. A deviation of ± 1.0 LU is acceptable for parts where an exact normalization to the target level of -23.0 LUFS is not practically achievable, due to the dynamic content of games. The loudness levels of a game part are often unpredictable. When a part consists of only background elements (for example, the background music), or where parts are deliberately mixed lower, the loudness level may lie outside the tolerance.

The Sony Worldwide Studios Audio Standards Working Group (ASWG) recommends in ASWG-R001 the following elements:

- The Average Loudness Level of audio content within a title developed for home-based platforms should be normalized to a Target Level of -23 LUFS, with a tolerance of ± 2 LU, and this tolerance is acceptable considering the non-linear nature of interactive entertainment audio content;
- The Average Loudness Level of audio content within a title developed for portable platforms should be normalized to a Target Level of -18 LUFS, with a tolerance of ± 2 LU;
- Audio content should be measured as a whole, and should not focus on dialogue, sound effects or music specifically;
- Audio content should be measured for as long as is practical and for a minimum of 30 minutes, and the measured sections of any title should be a representative cross section of all different parts of any title, in terms of game play;

More on Loudness Range (LRA)

The recommended maximum LRA is dependent on the genre, the targeted audience, or the environment where the game is intended to be played. The maximum LRA value may also be different for distribution platforms such as the mobile game. Here are a few examples of loudness ranges depending on the listening environment:

Environment	LRA
Home Theater	20
Living Room	18
Kitchen	15
Living Room (Late night)	9
Public Transportation (mobile)	6

As a general rule:

- A noisy environment requires a lower LRA so that the listener can hear all content correctly
- Louder loudspeakers allow a greater LRA

Loudness Range is a measure that also helps to decide if dynamic compression is necessary.

EBU suggests that with the measure of Loudness Range, it is possible to determine appropriate measures for potential dynamic compression of a program to fit within the tolerance window of the audience or distribution platform. In practice, overall low-level compression may lead to satisfactory results and ensure uniform compression of the whole signal range:

- a low threshold (< -40 dBFS)
- a moderate compression ratio (1:1.2 – 1:1.5)
- a long release-time (>1 s)

Again, EBU suggests that depending on the original loudness level, a shift to the target level of -23 LUFS may be performed in parallel through adjusting the make-up gain of the compressor accordingly.

More on True Peak

The True Peak measurement detects analog clipping. Even when the peak lies in between samples, so that the distortion that can happen in subsequent Digital-to-Analogue converters, sample rate converters, or commonly used codecs can be predicted and avoided.

The Maximum Permitted True Peak Level recommended in EBU R 128 is -1 dBTP.

To monitor the true peak on the Master Audio Bus:

1. Open one of four Meter views, using menu: **Views > Meter > Meter - Sync Group 1...4**
2. Click [...] and select the **Master Audio Bus**
3. From the drop down, select **True Peak** to monitor the True Peak

Measuring the Loudness in Wwise

To monitor the loudness on the Master Audio Bus:

1. Open one of four Loudness Meter views, using menu: **Views > Loudness Meter > Loudness Meter - Sync Group 1...4.**
2. Click [...] and select the **Master Audio Bus.**
3. Click **Capture** to start calculating the Integrated loudness.

To capture the loudness level

1. Open the **Loudness Meter** view.

2. In the title bar of the view, select a sync group from the sync group selection icon.
3. Select a bus to measure the loudness by clicking the [...].
4. On the Wwise Toolbar, click **Capture**.
Note: Do not click Capture in the **Loudness Meter View**.
5. Open the **Performance Monitor** (or press **F6** for the profiler layout, or **Shift-N**)
6. On the **Performance Monitor** toolbar, click the view settings icon.
7. Enable one or many of the loudness values available in the list, corresponding to the sync group you selected first (for example, *Momentary Loudness Level - Sync Group 1*)
8. Click **OK**.

Getting to Know the Master-Mixer Console

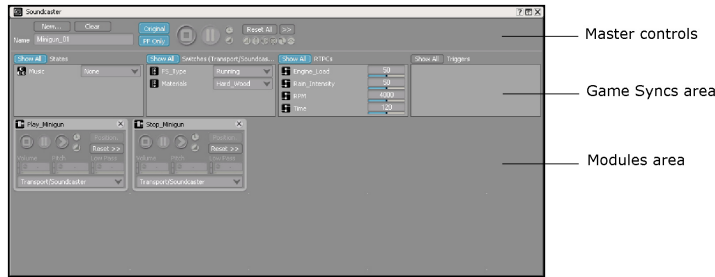
The Master-Mixer Console is a mixing console that groups a variety of bus properties into one view so that you can create the final mix of your game audio. When you are connected to a game, you can audition and then tweak the properties of your audio and motion as it is being played back in game in real time.

You can also monitor specific information related to the objects that are routed through the various busses in your project, including:

- **Meter** - Shows the peak levels for all channels.
- **Duck** - Indicates when a bus is ducked.
- **BG** - Indicates that a bus is tied to the background music option on the Xbox 360 or PlayStation 3 dashboards.
- **Bypass** - Indicates that the Bypass All option is selected for a particular bus.
- **Effect** - Indicates the particular effect or series of effects that have been applied to the bus.

The Master-Mixer console is a part of the Mixer layout. The Mixer layout groups the following views:

- **Master-Mixer Console** - Provides property, metering, mute/solo controls as well as information about ducking, bypassing, and effects. By double-clicking a selected bus, you can open the Audio or Motion Bus Property Editor to edit the settings information.
- **Soundcaster** - Provides a simulation environment where you can audition your audio and motion output. At any point in the game, you can use the Soundcaster to mix your audio and motion. For more information on using the Soundcaster, refer to [Chapter 31, *Creating Simulations*](#).



Note

The Mixer layout also includes the Event Viewer and the Project Explorer.

Fine-tuning your Mix using the Master Mixer Console

After you complete the routing for your project, you can create the final mix by connecting to your game and tweaking the relative properties in real time while the game is being played. You can use the Master-Mixer Console to easily modify the relative properties of each bus in your project. Relative properties are cumulative, which means that the values that you define will be added to the object property values.

While you are fine-tuning the mix, you can mute or solo one or more busses to help focus on one particular aspect of the sound or motion in your game or to troubleshoot specific problems.



Note

Changing the pitch of a bus will not affect any music objects routed through that bus.

To create the final mix using the Master-Mixer console:

1. Connect to the game by clicking the **Remote** button on the toolbar.

The Remote Connection dialog box opens with a list of consoles that are currently running on the network.

2. From the list, select the console to which you want to connect.



Note

The status of the console must be “Ready” before you can connect to it.

3. Click **Connect**.

Wwise connects to the remote game console. The name of the game console appears in the toolbar.

4. From the menu bar, click **Layouts > Mixer**.

The Mixer Layout is displayed.

5. With the game running, audition your audio or motion.

6. To adjust one of the following properties, type a value or drag the corresponding slider:

Volume - The level or amplitude of the audio output.

Pitch - The playback speed of the audio or motion output.



Tip

To mute or solo a particular bus, click the corresponding M or S button.

Related Topics

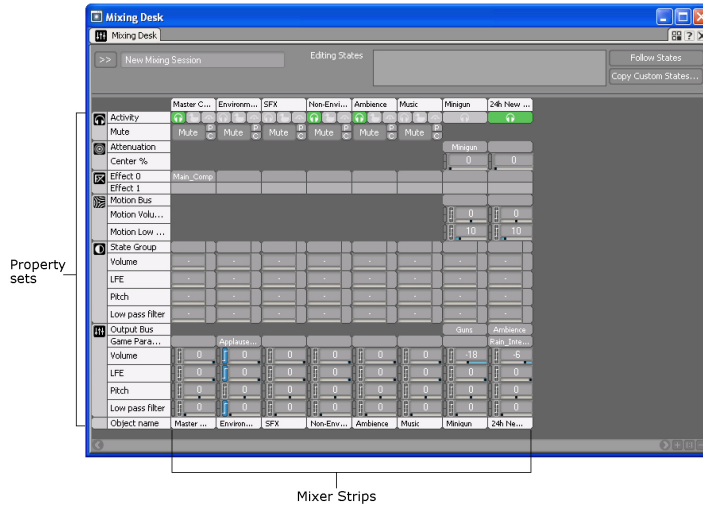
- [Getting to Know the Master-Mixer Console](#)
- [Applying Effects to Objects](#)
- [Defining the Relative Properties of a Bus](#)
- [Ducking Audio and Motion Signals](#)

Getting to Know the Mixing Desk

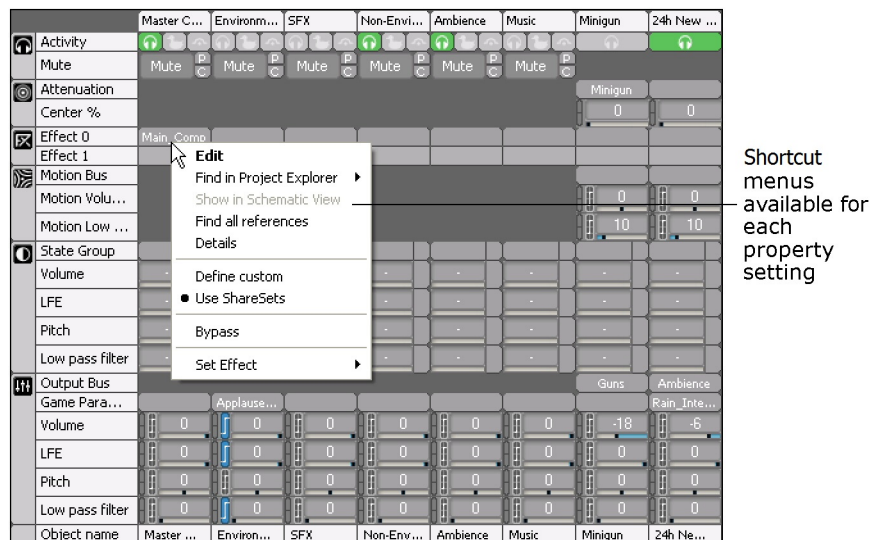
The Mixing Desk is a flexible and powerful mixing console that groups a variety of properties into one view allowing you to fine-tune the audio mix of your game in real time. You can add any object or bus to the Mixing Desk and then define object routing, apply effect and attenuation ShareSets, edit state properties, and modify the properties of individual objects and busses.

If you start a capture session, you can also view the activity of each object within the Mixing Desk, including when a voice is playing, if a bus is being ducked, and whether effects are being bypassed. Each audio object can also be muted or soloed allowing you to tweak the individual objects within your audio mix.

The Mixing Desk is basically a grid where each column is a mixer strip and each row is a section related to a set of common properties in Wwise. Each mixer strip is bound to an object, the name of which is displayed at the top and bottom of the each strip.



A shortcut menu is also available for each property setting in the mixer strip. These menus contain a set of commands specifically related to the selected property. For example, when you right-click one of the Effect slots (0-3), you can edit the properties of the inserted effect, set a new effect, bypass the effect, and so on. To access the shortcut menus, simply right-click the individual property settings in the mixer strip.




All elements within the Mixing Desk are saved within a Mixing Session. This allows you to create different mixing sessions for different components of your game. It also means that you can set up a mixing session and then continue to fine-tune the audio mix of your game at a later time.

Building a Mixing Session

Before you can use the Mixing Desk to fine-tune the mix of your game audio, you need to create and populate a mixing session. A mixing session is similar

to a preset or Soundcaster session in that it saves the contents of the Mixing Desk within your project so that you can go back to it at a later time. Each mixing session is saved on the Sessions tab of the Project Explorer, under its corresponding work unit.

To help you easily identify a mixing session in the interface, Wwise uses a unique icon to represent it.

Icon	Represents
	Mixing Session

Creating a Mixing Session

Before you can use the Mixing Desk, you need to create a mixing session. Mixing sessions contain a series of busses and/or objects that are arranged within a type of mixing console. A series of sliders and property value controls are displayed for each bus and/or object in the mixing session so that you can fine-tune the audio mix of your game..

You can create mixing sessions in the following two places in Wwise:

- The Sessions tab of the Project Explorer.
- From within the Mixing Desk.

To create a mixing session from the Project Explorer:

1. In the Project Explorer, switch to the Sessions tab.
2. Under the Mixing Sessions heading, do one of the following:

Select a work unit or virtual folder and then click the **Mixing Session** icon in the Project Explorer toolbar.

Right-click a work unit or virtual folder and click **New Child > Mixing Session** from the shortcut menu.

A new mixing session is added to the virtual folder or work unit.

3. Replace the default name with one that best represents the mixing session.
4. Double-click the new mixing session to load it into the Mixing Desk.

You are now ready to populate the mixing session with busses and objects.

To create a new mixing session from the Mixing Desk:

1. In the Mixing Desk, click the Mixing Session Selector button (>>) and select **New** from the list.

The New Mixing Session dialog box opens.

2. Select the work unit into which you want to create the new mixing session.
3. In the Name field, replace the default name with one that best represents the mixing session.
4. Click OK.

The mixing session is created and you are ready to start populating it with busses and objects.

Related Topics

- [Getting to Know the Mixing Desk](#)
- [Customizing the Information Displayed in the Mixing Desk](#)
- [Adding Objects/busses to a Mixing Session](#)
- [Reordering Mixing Strips within a Mixing Session](#)
- [Resizing the Mixing Strips within a Mixing Session](#)
- [Removing Mixing Strips from a Mixing Session](#)
- [Mixing Audio Using the Mixing Desk](#)

Adding Objects/busses to a Mixing Session

After you have created your mixing session, you can start populating it with the busses and/or individual objects within your project. When you add a bus or object to the Mixing Desk, a mixing strip is created. A mixing strip gives you an overview of the object's or bus' property settings. It also has a series of controls that allow you to monitor various activities, modify properties, set effects and states, and define positioning properties.



You can add any of the following objects to the Mixing Desk:

- busses
- Actor-mixers
- Containers
- Sounds
- Motion FX
- Music Containers
- Music Segments
- Music Tracks



Note

If a mixing session contains a mixing strip that relates to an object or bus that has been unloaded from the project, the mixing strip will be empty.

To add objects/busses to a Mixing Session:

1. Load a mixing session into the Mixing Desk by doing one of the following:
Click the Mixing Desk Selector (>>) and select a mixing session from the list.

- Double-click a mixing session from the Sessions tab of the Project Explorer.
- From the Project Explorer, drag a bus or object into the Mixing Desk.

A mixing strip is created in the Mixing Desk, representing the object or bus.

- Continue to add objects and/or busses to the Mixing Desk.



Note

When you have several mixer strips in the Mixing Desk, a blue insertion line displays the location where the mixing strip will be added.

Related Topics

- [Getting to Know the Mixing Desk](#)
- [Customizing the Information Displayed in the Mixing Desk](#)
- [Creating a Mixing Session](#)
- [Reordering Mixing Strips within a Mixing Session](#)
- [Resizing the Mixing Strips within a Mixing Session](#)
- [Removing Mixing Strips from a Mixing Session](#)
- [Mixing Audio Using the Mixing Desk](#)

Reordering Mixing Strips within a Mixing Session

After you have added busses and objects to a mixing session, you can arrange the mixer strips in any order by dragging them from one position to another.

To re-order the mixing strips within a mixing session:

- Load a mixing session into the Mixing Desk.
- Click the title bar of a mixing strip and then drag it to a new location in the Mixing Desk.

A blue insertion line displays the location where the mixing strip will be moved.

- Continue to re-arrange the mixer strips, as required.

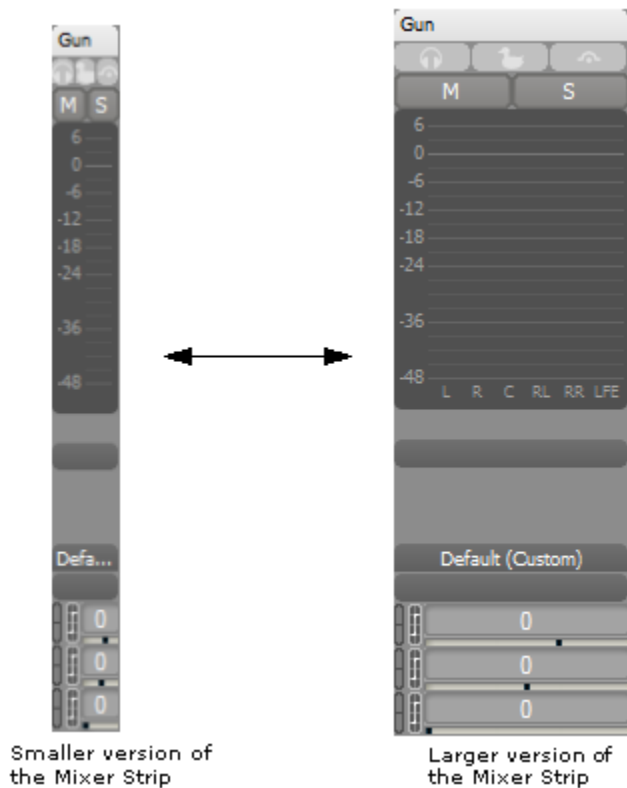
Related Topics

- [Getting to Know the Mixing Desk](#)
- [Customizing the Information Displayed in the Mixing Desk](#)
- [Creating a Mixing Session](#)

- [Adding Objects/busses to a Mixing Session](#)
- [Resizing the Mixing Strips within a Mixing Session](#)
- [Removing Mixing Strips from a Mixing Session](#)
- [Mixing Audio Using the Mixing Desk](#)

Resizing the Mixing Strips within a Mixing Session

If you find the mixing strips too small or too large, you can resize them.



To resize a mixing strip:

1. Load a mixing session into the Mixing Desk.
2. Click in the grid section of the Mixing Desk.
3. Do one of the following:

To make the mixing strips larger, press the **Ctrl** key while rolling the mouse's scroll wheel upwards or click the + sign next to the scroll bar.

To make the mixing strips smaller, press the **Ctrl** key while rolling the mouse's scroll wheel downwards or click the - sign next to the scroll bar.

Related Topics

- [Getting to Know the Mixing Desk](#)

- [Customizing the Information Displayed in the Mixing Desk](#)
- [Creating a Mixing Session](#)
- [Adding Objects/busses to a Mixing Session](#)
- [Reordering Mixing Strips within a Mixing Session](#)
- [Removing Mixing Strips from a Mixing Session](#)
- [Mixing Audio Using the Mixing Desk](#)

Removing Mixing Strips from a Mixing Session

At any point in your mixing session, you can remove one or more mixing strips from the Mixing Desk.

To remove a mixing strip from a mixing session:

1. Do one of the following:

Right-click the title bar of the mixing strip and select **Remove** from the menu.

Click the title bar of the mixing strip you want to remove and press the **Delete** key.

The mixing strip is removed from the mixing desk.

Related Topics

- [Getting to Know the Mixing Desk](#)
- [Customizing the Information Displayed in the Mixing Desk](#)
- [Creating a Mixing Session](#)
- [Adding Objects/busses to a Mixing Session](#)
- [Reordering Mixing Strips within a Mixing Session](#)
- [Resizing the Mixing Strips within a Mixing Session](#)
- [Mixing Audio Using the Mixing Desk](#)

Customizing the Information Displayed in the Mixing Desk

The Mixing Desk is customizable, which means that you decide what types of information will be displayed. Using the Mixing Desk Settings dialog box, you can add or remove different types of information, choose between property sliders or faders, and specify the height of the faders.

To customize the information displayed in the Mixing Desk:

1. Open the Mixing Desk by doing one of the following:

From the menu bar, click **Views > Mixing Desk**.

Press **Ctrl+Shift+M**.



2. In the Mixing Desk title bar, click the **View Settings** icon.

The Mixing Desk Settings dialog box opens.

3. Select the check box beside each type of information that you want displayed in the Mixing Desk.



Note

For a description of each option, click the Help icon in the Mixing Desk Settings dialog box.

4. If you chose to display the vertical faders, specify a height in the Vertical fader height text box.
5. Click **OK** to apply your settings.

Related Topics

- [Getting to Know the Master-Mixer Console](#)
- [Getting to Know the Mixing Desk](#)
- [Creating a Mixing Session](#)
- [Adding Objects/busses to a Mixing Session](#)
- [Reordering Mixing Strips within a Mixing Session](#)
- [Resizing the Mixing Strips within a Mixing Session](#)
- [Removing Mixing Strips from a Mixing Session](#)
- [Mixing Audio Using the Mixing Desk](#)

Mixing Audio Using the Mixing Desk

The Mixing Desk is the perfect tool to help you mix the different audio components of your game. You can connect to your game, start a capture session, and then monitor the activities of your audio while tweaking the individual properties for each object and bus within your mixing session in real time.

To mix audio using the Mixing Desk:

1. Load a mixing session into the Mixing Desk.
2. Connect to the game by clicking the **Remote** button on the toolbar.




The Remote Connection dialog box opens with a list of consoles that are currently running on the network.

3. From the list, select the console to which you want to connect.
4. Click **Connect**.

Wwise connects to the remote game console. The name of the game console appears in the toolbar.

5. Click the **Start Capture** button, if not already selected, to begin the capture session.

You can now monitor the following activities of the audio in your game:

Icon	Activity	Description
	Voice playback	Indicates when the object is being played. In the case of a bus, it indicates when a playing voice is being routed through the bus.
	Ducking	Indicates whether a bus is being ducked.
	Effect bypass	Indicates when an inserted effect is being bypassed.

6. With the game running, audition your audio.
7. Use the following tools to fine-tune the mix in your game:

Property sliders/faders - To modify the individual properties of the objects and busses in your mixing session, including volume, pitch, and low pass filter.

shortcut menus - To access a set of commands specifically related to the selected property. For example, when you right-click one of the Effect slots (0-3), you can edit the properties of the inserted effect, set a new effect, bypass the effect, and so on.

Follow states - To follow the state changes made in game.



Tip

At any time, you can double-click the name of the object, bus, state, or game parameter in one of the mixer strips to load the corresponding element into the Property Editor.

Related Topics

- [Getting to Know the Master-Mixer Console](#)
- [Getting to Know the Mixing Desk](#)
- [Customizing the Information Displayed in the Mixing Desk](#)

- [Creating a Mixing Session](#)
- [Adding Objects/busses to a Mixing Session](#)
- [Reordering Mixing Strips within a Mixing Session](#)
- [Resizing the Mixing Strips within a Mixing Session](#)
- [Removing Mixing Strips from a Mixing Session](#)

Chapter 30. Managing Platform and Language Versions

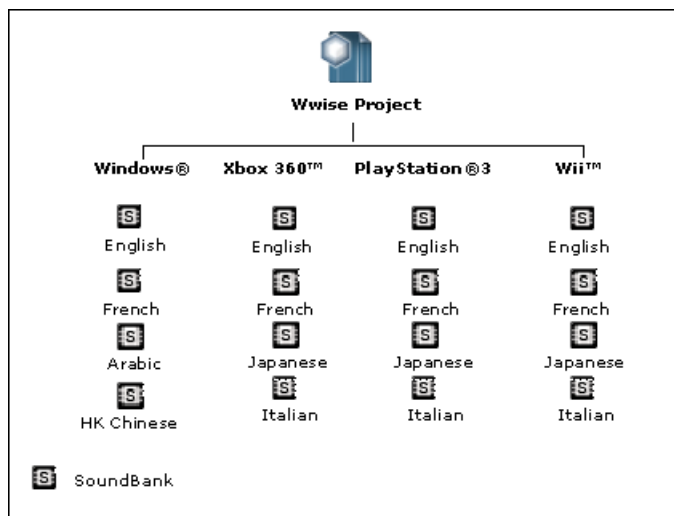
Overview	595
Authoring Across Platforms	595
Localizing Your Project	620
Versions Tips and Best Practices	628

Overview

Because game production is always a balancing act between optimizing quality and minimizing memory usage, you need to have the means to tweak and experiment with your properties and assets to deliver the best possible gaming experience within the constraints of each platform. In Wwise, you can customize assets, properties, and conversion settings per platform all within the same project. In addition, you can customize some language properties and conversion settings for the localization of your project. Using the simulation and profiling tools, you can verify your work to ensure that you have built successful language and platform versions.

When you have completed the audio production for each language and platform in your project, you can generate the individual SoundBanks for each simultaneously. For more information about working with SoundBanks refer to [Chapter 34, *Managing SoundBanks*](#).

The following illustration demonstrates how you can deliver different SoundBanks for the language versions of your audio for each platform.



Authoring Across Platforms

For audio designers, dealing with different platforms usually means creating different projects for each platform. In Wwise, this can all be done in one project. When you create your project, you are automatically ready to develop across platforms, even simultaneously if you want to. By default, all available platforms are active in a project, but you can remove a platform from a project to improve the efficiency of your workflow. For information on removing platforms from a project, refer to [Removing a Platform](#).

After you have created your project and imported your assets and music, you set up a series of conversion setting ShareSets that can be customized for each platform and, also, define how you want to use your assets based on the strengths and limitations of each platform.



Note

In Wwise, authoring across platforms implies that you have the rights to develop for the platforms. If you acquire additional platform rights, you need to update your installation package to include the added platforms.

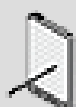
Wwise allows you to customize your project for each active platform using the following:

- [Converting Audio Files](#)
- [Customizing Object Properties per Platform](#)
- [Excluding Project Elements from a Platform](#)
- [Switching to a Different Platform](#)
- [Copying Settings from One Platform to Another](#)

Converting Audio Files

After you have imported your media files into Wwise, you are ready to convert them for your various platforms. Since many media files are likely to use the same conversion settings, you can create a series of conversion settings ShareSets and then assign these ShareSets to the various objects in your project. Like other properties in the project hierarchy, conversion setting ShareSets are inherited from parent to child. Of course, you can override these settings at any level in the hierarchy, which allows you to apply different conversion settings ShareSets to specific objects in the hierarchy.

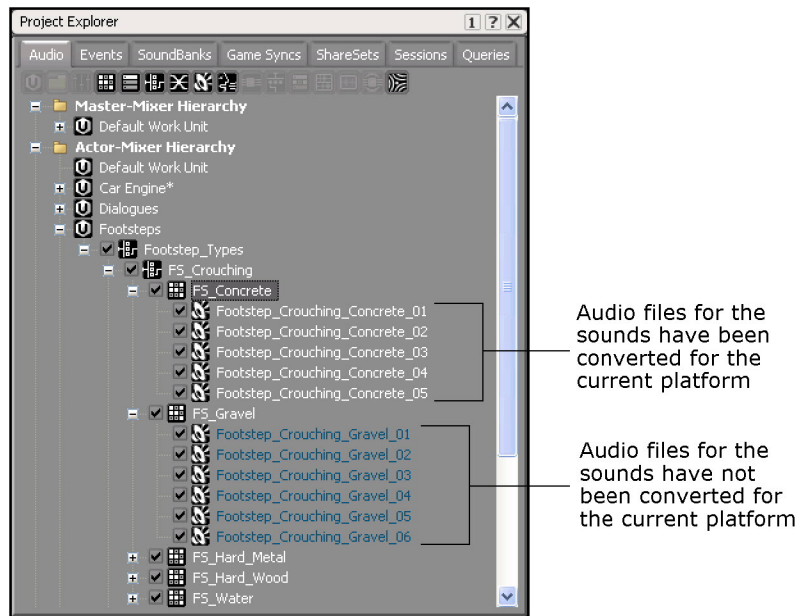
While you do not need to convert your media files to define properties and behaviors, it is a good idea to assign a conversion setting ShareSet to each object before you generate your SoundBanks to take advantage of the customized settings. If you don't assign a ShareSet to an object, Wwise will use the default conversion settings ShareSet. For more information on setting the default conversion settings ShareSet, refer to [Specifying the Default Conversion Settings](#).



Note

Wwise uses all processing cores on your machine to speed up the conversion process.

Sounds, music, or motion FX objects associated with sources that are not converted for the current platform appear blue in the **Audio** tab of the **Project Explorer**.



The conversion process includes three steps:

- [Creating Audio Conversion Settings ShareSets](#)
- [Assigning Conversion Settings ShareSets to Objects](#)
- [Converting Audio Files](#)

Creating Audio Conversion Settings ShareSets

The ShareSets you create to manage the conversion settings are based on the needs of your project and the requirements of each active platform. Many of your choices here can have a big impact on the performance and quality of your audio project. After applying the ShareSets to the objects in your project, you can go back and tweak your conversion settings ShareSets at any time to achieve the best possible quality within the constraints of the platform and the game. When you import audio files you can also speed up the process by re-using the same ShareSets that you have defined here for language and source versions.

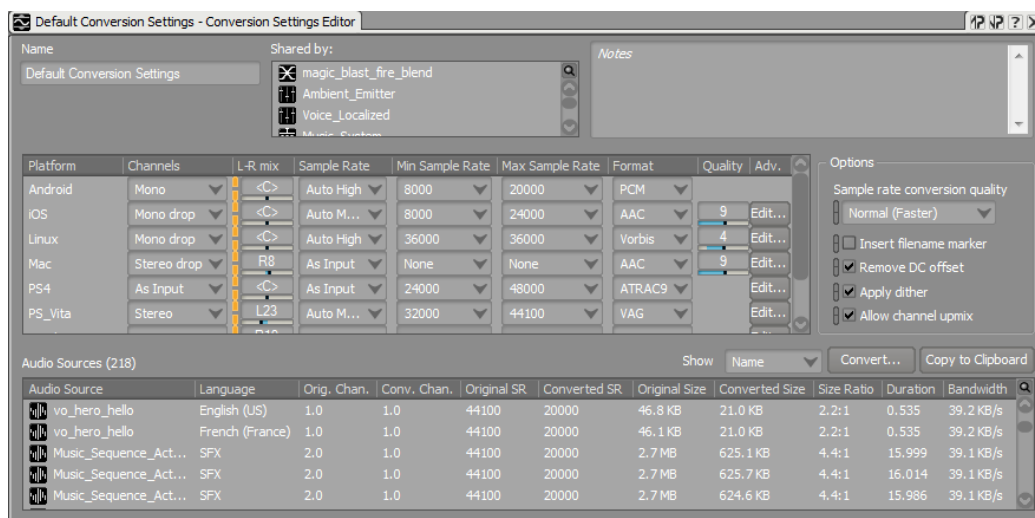


Note

You can use one of your ShareSets as the default conversion settings for your project. For more information on setting up the default conversion settings for your project, refer to [Specifying the Default Conversion Settings](#).

The Conversion Settings Editor is divided into two main sections:

- **Settings** - The area above the **Audio Sources**. It allows you to set the conversion settings for each platform, including sample rate, audio format, and number of channels.
- **Results** - The area listing all your audio sources. It allows you to compare the original and converted sources, including the number of channels, sample rate, and file size.



The audio conversion process retains the same pitch and duration as the original files; however, you can define the following properties for your conversion:

- Number of channels
- Left-Right mix
- Sample rate
- Audio format
- Audio format quality
- Sample rate conversion quality

You can also specify whether you want to:

- **Insert a filename marker** for lip-syncing or sub-titling;
- **Remove DC offset**;



Caution

We recommend not removing the DC offset for looping sounds. The remove mechanism is a high-pass filter, so there is no guarantee that it will modify the first and last sample of the loop in the same way because it does not know they will play

contiguously. This could create a discontinuity in the signal, which is audible as a click.

- **Apply dither;** or
- **Allow channel upmix,** which means mono source files will be converted to stereo when channels are marked as **Stereo** or **Stereo drop**.



Note

If this option is not selected, mono source files will remain mono regardless of the **Channels** settings.

About Audio Channels

When converting multi-channel audio sources, you must decide which channels to preserve. For more information, refer to [Channel Configuration](#).

In Wwise, you can choose from the following options:

Channel Option	Description	Comments
As Input	Preserves the same number of audio channels as the original media file.	Multi-channel files are not supported on platforms that do not natively support multi-channel outputs, such as the Wii, the 3DS™, the PS Vita, and iOS. The 'As Input' option is equivalent to the 'Stereo' option.
Mono	All channels are mixed into one.	The L-R Mix is only used for Stereo to Mono conversion. Other channel configurations are downmixed using the standard AC3 recipes (see Digital Audio Compression Standard) The LFE channel is always dropped.
Mono drop	All channels are dropped except the first one.	Depending on the channel configuration of the original file, the first channel can either be the Left or Center channel.
Stereo	All channels are mixed into front left and front right.	The L-R Mix is only used for Mono to Stereo conversion. Other channel configurations are downmixed using the standard AC3 recipes (see Digital Audio Compression Standard) The LFE channel is always dropped.
Stereo drop	All channels are dropped except channels that are defined as Left or Right.	In cases where no channel is defined as Left or Right, and there is a Center channel defined (mono), the following conversion occurs: Left = 0.707C Right = 0.707C The converted file will be twice the size of the original file.

It is important to note that Wwise does not do any multi-channel encoding; it simply feeds LPCM data to the console or system in either stereo, 5.1 or 7.1

surround. Once the LPCM data is received by the console or system, it can then be output in almost any format supported by the particular console or system, including Dolby, DTS, or DPL2. Some restrictions do exist, however, including:

- No support for 7.1 surround on the PlayStation 3.
- Support for stereo, and 5.1 and 7.1 surround on the Mac platform.
- Support for only stereo on the Wii, the 3DS, the PS Vita, and the iOS platforms.
- Other platforms such as Windows, Playstation 4 and Xbox One natively support up to 7.1 channels at their output, but Wwise is able to carry all standard channel configurations (up to 13.1 channels), as well as anonymous configurations (up to 256 channels). Note that these configurations require the use of a special sink plugin that can interpret them and/or pass them to dedicated hardware.

About Sample Rates

The sample rate determines the number of times per second a digital audio signal is sampled. When deciding on which sample rate to choose, many factors come into play and like other quality/performance issues, setting the sample rate is a balancing act. To give you as much control as possible, Wwise gives you many different sample rate conversion options:

- **As Input** - Converts the file using the same sample rate as the original file. If the sample rate is not available for a particular platform or audio format, then the closest sample rate available will be used.
- **Auto (Low/Medium/High)** - Converts the file using a sample rate selected by Wwise after performing an FFT analysis of the file. The difference between the low, medium, and high quality settings is the cutoff threshold value used by the algorithm. You can tweak the quality level of each setting by defining their threshold values in the **Project Settings** dialog box. For more information on the automatic sample rate detection performed by Wwise, refer to [Defining the Sample Rate Automatic Detection Settings](#).
- **300 to 48,000** - Converts the file using a specific sample rate. The sample rate range varies for each platform up to a maximum of 48,000 Hz. The exception is the Wii platform, where the maximum is 32,000 Hz, which is the default DSP mode for the Wii.



Note

In the cases **As Input** and **Auto**, you can further restrict the sample rate via the **Min Sample Rate** and **Max Sample Rate** combo boxes.

About Audio Formats

Before converting your audio files, you need to decide into which format they will be converted. Wwise supports multiple different audio formats to give you greater flexibility and control to work around the limitations of each platform. The different audio formats supported by Wwise are the following:

- **AAC** - A perceptual coding audio compression method for the Mac and iOS. AAC is said to achieve better sound quality than MP3 at similar bit rates. Compression is variable, content dependent, and the quality setting can be controlled by the “quality” slider. On iOS, AAC is decoded by the hardware-assisted codec *if it is available*. Note that the hardware may only decode one AAC sound at a time.



Note

Because of the presence of hardware support for AAC on iOS, this format is recommended for your background music on this platform. However, you should ensure that there is never more than one AAC sound playing at the same time. Additional AAC sounds are decoded in software, which uses a huge amount of CPU. Thus, you should avoid using this format with interactive music, since music segments usually overlap. Note that when using hardware-assisted decoding, the CPU usage displayed in the Wwise Profiler will appear to be very large. This is because decoding occurs within the audio processing thread of Wwise. On the other hand, most of this time is actually spent waiting for the hardware, during which the CPU is available to other threads of your game. Also, starting a new AAC voice may take a significant amount of time, and may even cause voice starvation on iOS. AAC has no limitation on looping, but it is not appropriate for sample-accurate playback. Only the following channel configurations are allowed (other configurations are automatically downmixed to stereo): mono, stereo, 0.1, 1.1, 5.0 and 5.1. Note that multichannel files and the LFE channel are not supported on iOS.

- **ADPCM** - An audio file conversion encoding method that quantizes the difference between a sound signal and a prediction that has been made of that sound signal. The ADPCM quantization step is adaptive, which results in better audio quality for the compressed signals. This method differs from PCM encoding where the signals are quantized directly.



Note

The ADPCM on the Wii platform uses the native hardware format (GCADPCM) and is decoded by the Wii DSP.

- [ATRAC9](#) - A perceptual encoding method for the PS Vita and PS4 that permits encoding of audio files at various bit rates while maintaining a very good perceived sound quality.
- [PCM](#) - An audio file conversion encoding method where distinct binary representations or pulse codes are chosen. These are then quantized by measuring values between two encoded points, selecting the value associated with the nearest point.
- [VAG](#) - An audio file conversion encoding method for the PS Vita based on ADPCM encoding. When using the hardware version of the Vita, this is your ADPCM option. VAG is not available for the software version of Vita because ADPCM is directly available to it.
- [Vorbis](#) - A perceptual encoding method that permits encoding of audio files at fixed and variable bitrates while maintaining a very good perceived sound quality. The balance between data compression efficiency and perceived sound quality is controlled using the Quality Factor setting or by specifying the maximum, minimum, and average bitrates per channel. The Vorbis encoder may require the use of a seek table. For more information, refer to [Using Seek Tables with the Vorbis Encoder](#).



Note

Audiokinetic's special implementation of Vorbis has been highly optimized for all platforms.

- [XMA](#) - A hardware supported perceptual coding audio compression method for the Xbox 360 and Xbox One. XMA is a console-optimized version of Windows Media Audio Pro. Compression is variable, content dependent, and the quality setting can be controlled by the **Compression Quality** slider. The most recent version, XMA 2.0, contains a new block size parameter for creating a seek table that facilitates seeking XMA data. The XMA audio format only supports mono and stereo files on the Xbox 360, but configurations up to 7.1 are supported on the Xbox One.
- [xWMA](#) - A software supported perceptual coding audio compression method for the Xbox 360. xWMA is a console-optimized version of Windows Media Audio Pro, and provides a greater compression ratio than XMA. Compression is variable, content dependent, and the quality setting can be controlled by the “bit rate” drop-menu. Note that xWMA has restrictions on looping and it only supports mono, stereo, and 5.1 files.



Note

Vorbis and XMA are transform codecs, which means that they can do seamless loops, but not sample-accurate loops. For both audio formats, the results will vary substantially depending on the material and original loop point position. XMA files are padded with silence at the beginning and end of the file. The latency resulting from this could be noticeable if the files are very small, and especially if they are played back to back. xWMA is also a transform codec, but looping in a region within xWMA sounds is not supported. On the other hand, it has a higher compression ratio than XMA, and smaller CPU usage than Vorbis. We recommend using Vorbis whenever you can, as its quality is much higher than XMA at the same bitrate, or using xWMA if looping is not required. If you run into problems related to sample-accuracy, you can try ADPCM, or even PCM. If you use too much CPU, you can try XMA or xWMA on the Xbox 360.

Each format has its advantages and disadvantages and the format you choose will depend on the CPU and memory restrictions of your particular game. For a further discussion on when to use which audio format, refer to [Versions Tips and Best Practices](#).

Due to the different specifications of each platform, not all formats are available on each one. The following table displays which audio formats are available by platform.

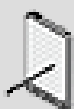
Platform	AAC	ADPCM*	ATRAC9	PCM	VAG#	Vorbis	XMA	xWMA
Android		✓		✓		✓		
iOS	✓	✓		✓		✓		
Linux		✓		✓		✓		
Mac	✓	✓		✓		✓		
Nintendo 3DS		✓		✓				
PS3		✓		✓		✓		
PS4		✓	✓	✓		✓		
Vita		✓	✓	✓	✓	✓		
Wii		✓		✓		✓		
Wii U		✓		✓		✓		

Managing Platform and Language Versions

Platform	AAC	ADPCM*	ATRAC9	PCM	VAG#	Vorbis	XMA	xWMA
Windows		✓		✓		✓		
Windows Phone		✓		✓		✓		
Xbox 360		✓		✓		✓	✓	✓
Xbox One		✓		✓		✓	✓	

* ADPCM on the Wii platform uses the native hardware format (GCADPCM) and is decoded by the Wii DSP.

VAG is applicable only to the PS Vita hardware format. If using Vita software format, ADPCM is used instead of VAG.



Note

Only PCM, ADPCM, and Vorbis audio formats support all configurations of multi-channel files. When the channel configuration is not supported by the format, Wwise downmixes it to the next supported configuration.

About DC Offsets


It is a good idea to remove the DC offset using a DC offset filter because DC offsets can affect volume and cause artifacts in Wwise. There are some cases, however, where you should not remove the DC offset, for example, for sample accurate containers. In other cases, for example, where sounds are normalized to 0 dB, you may or may not need to remove the DC offset. During the conversion process, DC offsets are removed by default. You can, however, disable this setting if needed in the Conversion Settings dialog box.



Caution

If you are generating motion directly from your audio sources, you should be aware that the removal of the DC offset may alter the motion output for controllers.

To create a conversion settings ShareSet:

1. In the Project Explorer, switch to the ShareSets tab.
2. In the Conversion Settings section, select the work unit into which you want to create the new ShareSet.
3. From the Project Explorer toolbar, click the **Conversion Settings** icon .

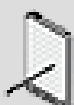
A new conversion settings ShareSet is created in the selected work unit.

4. Name the ShareSet appropriately and press **Enter**.
5. Double-click the new ShareSet to load it in the Conversion Settings Editor.
6. Specify the Channels for each platform by selecting one of the following:
 - **As Input** - To preserve the same number of audio channels as the original media file.
 - **Mono** - To mix all channels into one mono channel.
 - **Mono drop** - To drop all channels except the first one.
 - **Stereo** - To mix all channels into the front left and front right.
 - **Stereo drop** - To drop all channels except the ones defined as Left and Right.



Note

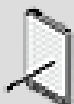
By default, the channel settings are linked across all platforms. To specify a unique channel setting for a particular platform, unlink the property first and then define the setting.



Note

If you don't want to increase the number of channels for mono files, make sure to disable the **Allow channel upmix** option.

7. If you are converting a stereo source to mono or vice versa, you can use the L-R Mix settings to specify the power level of the signal assigned to the left and right channels.



Note

By default, the L-R Mix settings are unlinked across all platforms. To specify a common L-R Mix setting for particular platforms, link the property first and then define the setting.

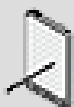
8. From the Sample Rate list, select the frequency with which the audio files will be sampled per second during conversion. Depending on the particular circumstances of your game, you can select one of the following options:

As Input - To convert the file using the same sample rate as the original file. If the sample rate is not available for a particular platform or audio format, then the closest sample rate available will be used.

Auto (Low/Medium/High) - To convert the file using a sample rate selected by Wwise after performing an FFT analysis of the file. The difference between the low, medium, and high quality settings is the cutoff threshold value that identifies the frequency used to determine the best sample rate in which to convert your files. You can tweak the quality level of each setting by defining their threshold values in the Project Settings dialog box. For more information on the automatic sample rate detection performed by Wwise, refer to [Defining the Sample Rate Automatic Detection Settings](#).

300 to 48,000 - To convert the file using a specific sample rate. The sample rate range varies for each platform up to a maximum of 48,000 Hz; however, Nintendo platforms - Wii, Wii U, and 3DS - have a maximum of 32,000 Hz.

9. If the **Sample Rate** is set to either **As Input** or **Auto**, then use the **Min Sample Rate** and **Max Sample Rate** entries to restrict the conversion sampling rate.
10. Specify the audio format for the conversion by selecting one of the following: [AAC](#), [ADPCM](#), [ATRAC9](#), [PCM](#), [VAG](#), [Vorbis](#), [XMA](#), or [xWMA](#).



Note

Click the **Edit** button to modify the encoding parameters for the ADPCM, Vorbis, XMA, xWMA, and AAC audio formats. For a complete description about the encoding parameters for these audio formats, click the **Help** button in the corresponding dialog box. For information on best practices for selecting parameters for the Vorbis audio format, refer to the Vorbis Encoder Parameters page in the reference documentation.

11. From the Sample rate conversion quality list, select the method that will be used to convert the file's sample rate. You can select either of the following options:
 - **Normal (Faster)** - Produces a good quality conversion that is three to six times faster than the Best option.
 - **High (Slower)** - Produces the best quality conversion.



Note

If you expect your content to contain high frequencies and you are converting to sample rates below 24 kHz, it is recommended that you use the High option.

12. If you want a marker to be created at the beginning of each converted file, select **Yes** from the Insert Filename Marker list.

The marker will only contain the filename and not the file's path and extension. Having the name visible can be useful when you want to bind an action to a sound playing in the sound engine, for example, when lip-syncing or sub-titling.

13. If you don't want DC offset to be removed during the conversion process, clear the **Remove DC Offset** check box.

By default this option is selected. In most cases, it is preferable to remove any DC offset. There are cases, however, where the DC offset may not need to be removed, including:

Sounds that will be added to sample accurate containers.

Sounds that are normalized to 0 dB.

For more information about how DC offsets affect audio signals in Wwise, refer to [Removing DC Offsets](#).



Caution

If you are generating motion directly from your audio sources, you should be aware that the removal of the DC offset will alter the motion output.

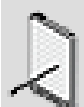
14. If you don't want dithering to be applied during the bit rate conversion, clear the **Apply Dither** check box.

Dither is the noise added to a signal prior to quantization in order to reduce the distortion and noise modulation that results from the quantization process. Dithering is only applied when the resolution changes, for example, from 24 bits to 16.

15. Close the Conversion Settings Editor.

The settings that you have specified are automatically saved and the ShareSet can now be assigned to one or more objects in your project hierarchy.

16. Repeat steps 1-14 to create as many conversion settings ShareSets as needed for your project.



Note

Before the Audio Sources table is populated, you must assign the conversion settings ShareSet to an object and then convert the audio files using these settings. For more information on

assigning a conversion settings ShareSet to an object, refer to [Assigning Conversion Settings ShareSets to Objects](#).

Related Topics

- [Copying Settings from One Platform to Another](#)
- [Assigning Conversion Settings ShareSets to Objects](#)
- [Converting Audio Files](#)
- [Customizing Object Properties per Platform](#)
- [Selecting Sources per Platform](#)
- [Excluding Project Elements from a Platform](#)
- [Switching to a Different Platform](#)

Removing DC Offsets

The *DC offset* represents the percentage away from zero amplitude that a signal lies. In most cases, because it is problematic for sounds, the DC offset should be removed. If you choose to maintain the offset, you should be aware of some of the undesirable effects in Wwise:

- A normalized sound with a DC offset will not reach its highest volume during audio processing because the offset consumes the *headroom*. This problem can extend to the mix as a whole because mixing two sounds where one is with a DC offset and the other is not, will cause the them both to have a DC offset.
- Any signals with recursive DSP algorithms applied are likely to be very sensitive to audio files with DC offsets and this will cause artifacts. This is especially the case for *reverb* plug-ins, as well as Environmental FX plug-ins.



Caution

We recommend not removing the DC offset for looping sounds. The remove mechanism is a high-pass filter, so there is no guarantee that it will modify the first and last sample of the loop in the same way because it does not know they will play contiguously. This could create a discontinuity in the signal, which is audible as a click.

To avoid unoptimized performance issues, the DC offset is not removed at run-time. Consequently, it is recommended to remove the DC offset before importing the files into Wwise, which can be done using a DC offset removal filter (available in most audio authoring tools) or during the conversion process in Wwise (in the **Conversion Settings** window).

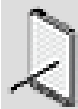
Assigning Conversion Settings ShareSets to Objects

After you have created the conversion settings ShareSets for your project, you can start assigning them to the objects in your project hierarchy. Conversion settings ShareSets, like other object properties, are inherited from parent to child, which means that if you assign a conversion settings ShareSet to an actor-mixer, all containers and objects below it will automatically use the same ShareSet. Of course, you can override the ShareSet used by the parent object, so that you can apply a different conversion settings to specific objects and/or sources in the hierarchy.

By default, all conversion settings instances you create begin as ShareSets. However, you may want tweak a ShareSet for one particular object. If you don't want to share the conversion settings between objects, you can create a custom instance of the ShareSet. When using a custom instance of a ShareSet, the settings are applied to the current object only. If any changes are made to the conversion settings only this object will be affected.

To assign a conversion settings ShareSet to an Object:

1. Load a top-level object into the Property Editor.



Note

If the object is not a top-level object, you must select the Override parent option before you can set the Conversion Settings options.

2. Switch to the **Conversion Settings** tab.
3. In the Conversion Settings group box, click the Selector button (>>).

A list of conversion settings ShareSets that are currently available in the project is displayed.

4. From the list, select the ShareSet you want to apply to the object.
5. If you want to apply a custom instance of the ShareSet to the object, select **Define custom** from the Mode list.

The name of the conversion settings instance now has the word “Custom” appended to it. From now on, changes you make to the instance will only affect the one object that uses it.

6. Click the **Edit** button to load the ShareSet or custom instance into the Conversion Settings Editor, where you can fine-tune the various conversion settings.
7. In the Audio Sources table, review the list of audio sources to which these conversion settings will be applied to make sure that it is correct.

To complete the information in this table, you must convert the audio sources. For more information on converting audio sources, refer to [Converting Audio Files](#).

Related Topics

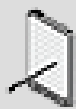
- [Creating Audio Conversion Settings ShareSets](#)
- [Converting Audio Files](#)
- [Customizing Object Properties per Platform](#)
- [Selecting Sources per Platform](#)
- [Excluding Project Elements from a Platform](#)
- [Switching to a Different Platform](#)

Converting Audio Files

After you have created your conversion settings ShareSets and applied them to the various objects in your project, you are ready to convert your audio files. You can decide to convert only the files for a selected object in the hierarchy, or all unconverted audio files. You can also specify the scope for this conversion based on:

- **Platforms** - The current or all of the platforms in the project.
- **Languages** - The current or all of the languages created for the project. This converts the sound voice files only.
- **Sources** - The In Use version of the sound in the Contents Editor or all versions (sources) of a sound.

Converting audio files can be a time-consuming process, especially when you have thousands of assets in your project. To help speed up this process, Wwise automatically uses all processing cores on your machine.



Note

If you experience problems converting your audio sources, it may be due to the length of your filename. For more information on filename limitations, refer to [Understanding File Length Limitations in Wwise](#).

To convert audio files for a specific object:

1. To open the Audio File Conversion dialog box, do one of the following:
 - In the Audio tab of the Project Explorer or Contents Editor, right-click the object you want to convert, and from the menu, select **Convert**.

- Select an object or audio source and then from the menu bar, click **Edit > Convert**.
 - On the Edit tab of the SoundBank Editor, select one or more objects, right-click, and select **Convert** from the menu.
 - In the Conversion Settings Editor, select one or more audio sources in the Audio Sources table and click **Convert**.
 - The Audio File Conversion dialog box opens.
2. In the Platforms group box select one or more of the listed platforms.
 3. In the Languages group box, select one of the following options:
 - **Current language**
 - **All languages**
 4. In the Sources group box, select one of the following options:
 - **In Use versions**
 - **All versions**
 5. Click **OK**.

The Conversion - In Progress dialog box opens. The first progress bar displays the overall progress of the audio source conversion process. One or more secondary progress bars display the progress of individual files as they are being converted. The number of secondary progress bars will depend on the number of processing cores on your machine.

When the file conversion is complete, the dialog box closes.



Note

If Wwise encounters any issues while converting the audio source, it will display information about them in the conversion log at the bottom of the dialog box. You can stop the conversion process at any time to investigate these error messages further by clicking **Stop**.

6. In the Conversion Settings Editor, you can review the information in the Audio Sources table to make sure that each converted source has the correct number of channels, is in the correct format, uses an appropriate sample rate, and is a reasonable size.



Note

To view the converted results information for a different platform, simply select the platform from the Platform Selector list in the Wwise toolbar.

To convert all audio files in the Audio tab:

1. From the Project menu, select **Convert All Audio Files**.

The Audio File Conversion dialog box opens.

2. In the Platforms group box select one or more of the defined platforms.
3. In the Languages group box, select one of the following options:

Current language

All languages

4. In the Sources group box, select one of the following options:

In Use versions

All versions

5. Click **OK**.

The Conversion - In Progress dialog box opens. The first progress bar displays the overall progress of the audio source conversion process. One or more secondary progress bars display the progress of individual files as they are being converted. The number of secondary progress bars will depend on the number of processing cores on your machine.

When the file conversion is complete, the dialog box closes.



Note

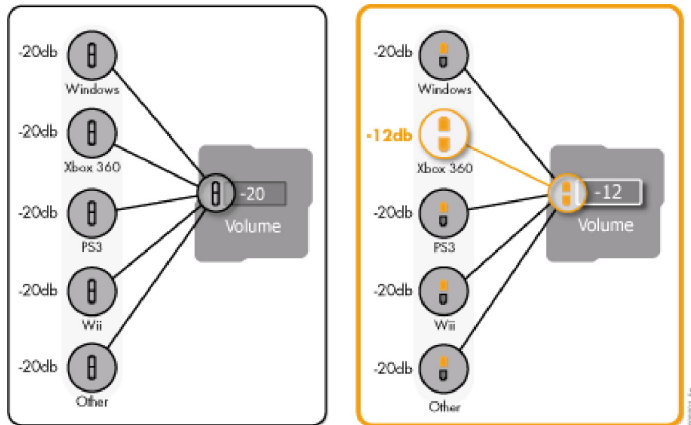
If Wwise encounters any issues while converting the audio sources, it will display information about them in the conversion log at the bottom of the dialog box. You can stop the conversion process at any time to investigate these error messages further by clicking **Stop**.

Related Topics

- [Creating Audio Conversion Settings ShareSets](#)
- [Assigning Conversion Settings ShareSets to Objects](#)
- [Customizing Object Properties per Platform](#)
- [Selecting Sources per Platform](#)
- [Excluding Project Elements from a Platform](#)
- [Switching to a Different Platform](#)

Customizing Object Properties per Platform

In Wwise, when you define your sound or motion properties for a particular platform in your project, these properties are set across all active platforms by default. These properties are said to be linked across platforms. This streamlines creating projects across platforms. But, if you want to customize the properties for a specific platform you can unlink its properties, and define the properties that you want. When you unlink properties, the link indicator becomes orange.



Properties are linked across all active platforms.

Volume is unlinked and customized for "XBox 360" platform, and the remaining platforms are partially unlinked.

When you are defining the properties for your platforms, you can easily tell if the properties are unlinked on another platform. The link/unlink indicator will be partly orange indicating that it is partially unlinked.

To unlink a property for a platform:

1. Right-click the link indicator beside the property that you want to unlink.

The shortcut menu is displayed.

2. Click **Unlink**.

The indicator turns orange and the values that you define will only apply to the current platform. When you switch platforms, the indicator will display that this property is unlinked on another platform.

When you switch platforms, the indicator will display that this property is unlinked on another platform.



Note

To link the properties again, right-click the property indicator and select link in the shortcut menu.

Related Topics

- [Copying Settings from One Platform to Another](#)
- [Creating Audio Conversion Settings ShareSets](#)
- [Converting Audio Files](#)
- [Selecting Sources per Platform](#)
- [Excluding Project Elements from a Platform](#)
- [Switching to a Different Platform](#)

Selecting Sources per Platform

In Wwise, a sound or motion object can have multiple sources. By default, a sound or motion object uses the same source when it is played across platforms. However, you can specify different sources for each platform by unlinking the sources in the Contents Editor. The source that you have selected will be used when the SoundBank is generated for that platform.

This sound object's audio source has been unlinked for the current platform.

This source is being used on the current platform.

Name	Audio File	Vol.	Offset	Duration	Notes
PF - English (US)	Use				Add Source >>
enter01	enter01.wav	0	0	1.114	
enter01_low	enter01.wav	0	0	1.114	
Wwise Silence	Use	1.5	0	0	
PF - French (Canada)	Use				Add Source >>
enter01	enter01.wav	0.3	0	1.114	
Wwise Silence	Use	1	0	0	



Note

For voice sound objects, you can unlink the source Use property separately for each language in your project. The same is true for the different sources created for the Controller motion device.

For example, let's say the characters in your fighting game have been given different names depending on which platform the game will be played. The hero of your game was originally named Max, but that character has been

renamed Arthur on the PlayStation 3. Certain voice sound objects would therefore contain two different audio sources, each a nearly identical line of dialogue mentioning a different name. You could unlink the sound object Use property in the PlayStation 3 version so that the sound object in that version would use Arthur's lines instead of Max's.

When you are defining which sources will be used for your platforms, you can use the link/unlink indicator to check each source's status. When a source is unlinked on the current platform, the link/unlink indicator is completely orange. However, when the source is unlinked on another platform, the indicator is partly orange.

To select a source for a platform:

1. In the Contents Editor, select the source that you want to use by clicking the corresponding Use option.

The selected source will be used across all platforms.

2. If you only want to use this source on the current platform, right-click the Use link indicator in the title bar.

The shortcut menu is displayed.

3. Click **Unlink**.

The indicator turns orange and the audio source you decide to use will only be used on the current platform.



Note

To link the source Use property again, right-click the property indicator and select **Link** from the shortcut menu.

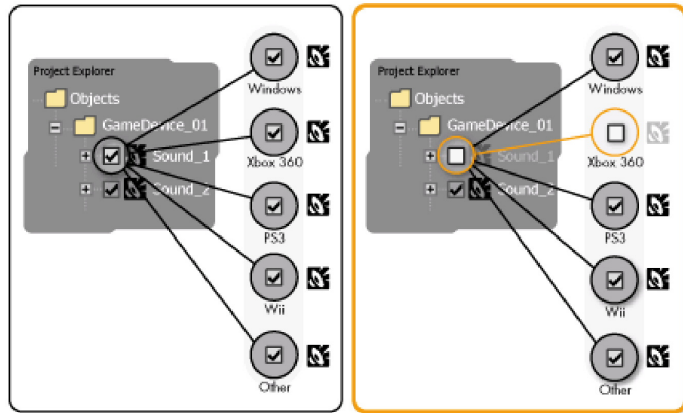
Related Topics

- [Copying Settings from One Platform to Another](#)
- [Creating Audio Conversion Settings ShareSets](#)
- [Converting Audio Files](#)
- [Customizing Object Properties per Platform](#)
- [Excluding Project Elements from a Platform](#)
- [Switching to a Different Platform](#)

Excluding Project Elements from a Platform

When you are developing your audio and motion, you need to keep in mind the memory requirements for the game and for the game platform. You also need

to consider which platforms can adequately handle certain assets. To optimize each platform you may need to include or exclude certain sounds or motion objects. When you have excluded an object, that object is not included in the SoundBanks that are generated for that platform.



Sound_1 is included in all platforms

Sound_1 is excluded from the Xbox 360 platform

You can decide to include or exclude a specific object, folder, event action, or language from a platform using the platform check boxes in many different views in Wwise, including the following:

- Project Explorer
- Property Editor
- Contents Editor
- Event Editor

After you have decided which objects to include or exclude, you can audition your platform-specific audio or motion in the Transport Control. You can also use the Soundcaster and the Game Profiler to help decide which objects to include or exclude in a platform. For more information about profiling and simulations, refer to the following sections:

- [Chapter 33, Profiling](#)
- [Chapter 31, Creating Simulations](#)

To exclude a project element from the current platform in a Wwise view:

1. In one of the Wwise views with the Platform check box, clear the Platform check box.

The project element is no longer included in the current platform.



Tip

You can also include/exclude selected project elements from one or all platforms using the shortcut menu options in the Audio tab of the Project Explorer.

To exclude project elements from all platforms:

1. On the Audio tab of the Project Explorer, select the objects or folders that you want to exclude from all platforms.
2. Right-click the selection and select **Platform > Exclude Selection - All Platforms**.

The check box beside each object and/or folder is cleared and they are now excluded from all platforms.



Tip

You can also use the shortcut menu to exclude/include objects and folders from the current platform, and include objects and folders for all platforms.

Related Topics

- [Copying Settings from One Platform to Another](#)
- [Creating Audio Conversion Settings ShareSets](#)
- [Converting Audio Files](#)
- [Customizing Object Properties per Platform](#)
- [Selecting Sources per Platform](#)
- [Switching to a Different Platform](#)

Switching to a Different Platform

Since your Wwise project contains all active platform versions, you can easily switch from one platform to another at any point in the development cycle. For example, you might be developing on more than one platform simultaneously, and you want to systematically define a specific object property for each platform. After you have switched platforms, the current platform version is displayed.

To switch between active platforms:

1. On the toolbar, click the arrow to display the Platform Selector list.

2. From the Platform Selector list, select the new platform.

The platform version that you have selected is displayed.



Note

Only platforms added to the project are displayed in the Platform Selector list. For information on adding a platform, refer to [Adding, Removing, and Copying Platforms](#).

Related Topics

- [Creating Audio Conversion Settings ShareSets](#)
- [Converting Audio Files](#)
- [Customizing Object Properties per Platform](#)
- [Selecting Sources per Platform](#)
- [Excluding Project Elements from a Platform](#)

Copying Settings from One Platform to Another

Since Wwise allows platform-specific editing in your project, you may, at some point, want to copy these settings from one platform to another. For example, if you have been working on two platforms for some time and are now ready to add a third one, you might want to start with a "copy" of one of the platforms you have already customized.

The **Copy Platform Settings** dialog box, accessed through the [Platform Manager](#), lets you do just that. Simply specify the source and target platforms, and Wwise will copy over all platform-specific settings from the source to the target platform.



Note

Platform settings are distinct from the **Base Platform**. Once the latter is selected for a platform, it does not change. If you need a new **Base Platform**, then you will have to create a new platform.

Platform-specific settings that are copied over from the source to the target platform

- Unlinked object parameters



Note

There are a few exceptions: platform-specific settings from the **SoundBanks** and **External Sources** tabs within the **Project Settings** dialog are not copied to the target platform.

- Platform inclusion/exclusion
- Unlinked active source (specified in the Contents Editor for Sounds and Motion FX)
- Conversion settings



Note

If a conversion format used on the source platform is not supported on the target platform, the default format for the target will be used instead, and a warning will appear in the log. Also, if the sample rate specified on the source platform is not supported by the format on the target platform, it will be fixed and a warning will appear in the log.

- Unlinked 3D Attenuation (specified in the Positioning tab of the Property Editor)
- Unlinked 3D Attenuation curves (defined in the Attenuation Editor)
- Unlinked Effects (specified in the Effects tab of the Property Editor)



Note

If an effect used on the source platform is not supported on the target platform, no effect will be set for the corresponding effect ID on the target platform.

- Unlinked Dialogue Event path target object

When a setting is unlinked on the source platform, it is unlinked and copied to the target platform. When a setting is linked on the source platform but unlinked on the target platform, it is relinked on the target platform.

Related Topics

- [Overview](#)
- [Specifying Volume Thresholds for a Project](#)
- [Customizing Object Properties per Platform](#)

- [Excluding Project Elements from a Platform](#)
- [Selecting Sources per Platform](#)
- [Creating Audio Conversion Settings ShareSets](#)
- [Applying an Attenuation ShareSet to an Object](#)
- [Defining the Attenuation Curves for Various Object Properties](#)
- [Using Effects](#)
- [Assigning Objects to Paths](#)

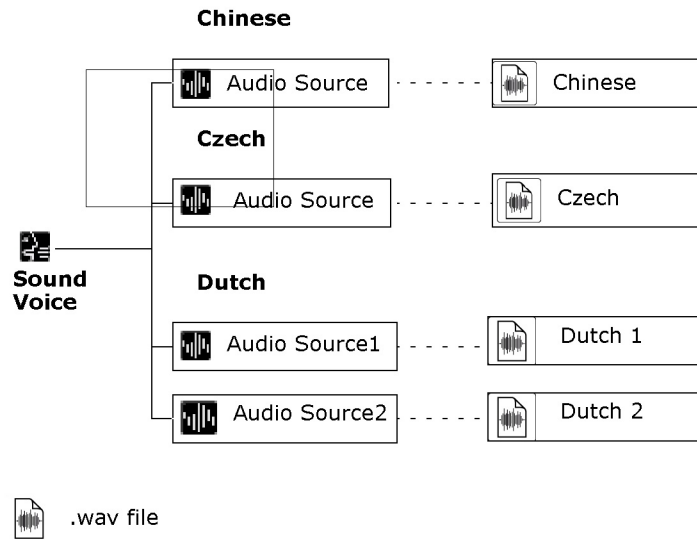
Localizing Your Project

A good localized project does not just involve replacing Sound Voice objects with their translated versions. The localized version must successfully recreate the same audio experience that has been developed in the original language. Technically, this involves tweaking and syncing the different language versions' properties so that the quality of the localized project matches the original version. Depending on the release schedule for the game, localization might be ongoing during development, or it may occur after the game is complete. In Wwise you can localize at any point in the development cycle. Just as you can author on multiple platforms simultaneously in one project, you can also create several language versions in one project and optimize and edit each language version in Wwise.

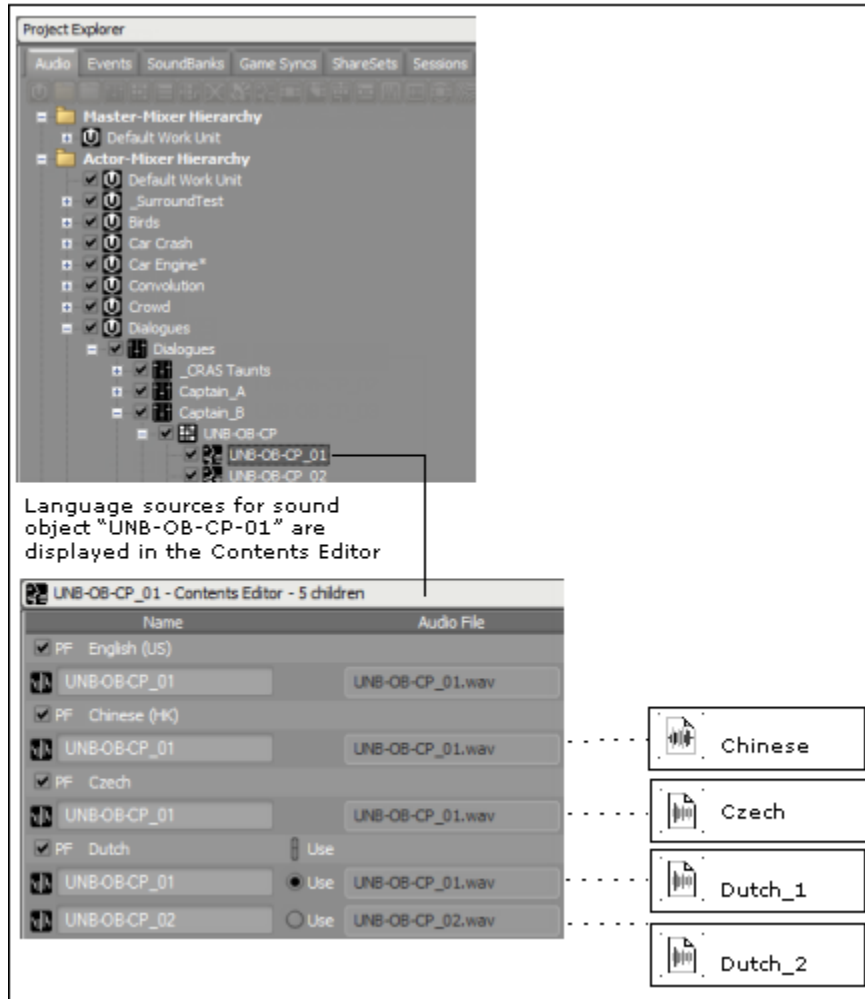
In Wwise, managing the localization of your project involves the following tasks:

- Create the languages for the project.
- Define the reference language.
- Import language files.
- Audition and profile the language versions.

In Wwise, the language versions are stored as sources of the Sound Voice object and are associated with different audio files for each source. The following illustration demonstrates the relationships between the sound object, its language sources, and the language files.



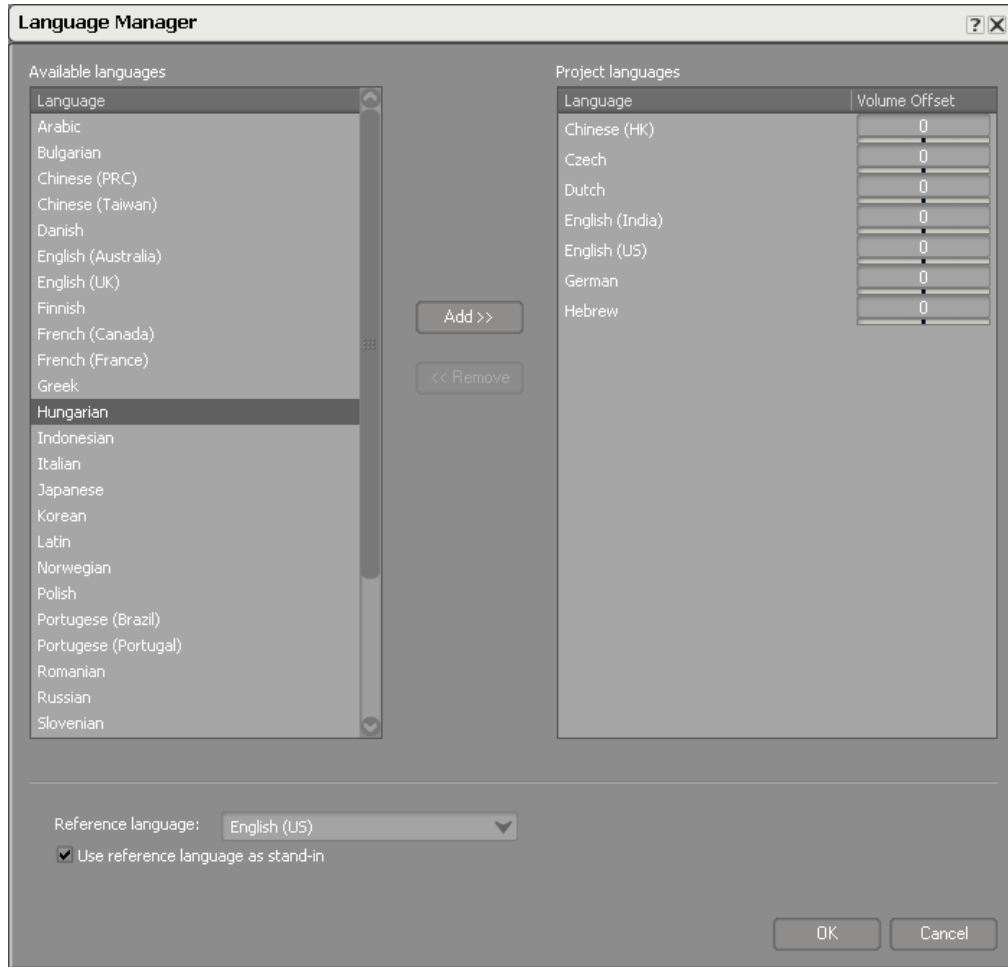
The Sound Voice object contains the sources for each language, but they are only displayed in the Contents Editor.



Managing the Languages for Your Project

After you have determined which languages that you are localizing, you can create them for your project. The project languages are displayed in the Language Selector on the toolbar and in the Contents Editor for Voice objects. Since not all the languages files will be available when you are creating your list, you can choose a reference language as a substitute for languages that are not yet ready. The conversion settings associated with the reference language can also be used as default conversion settings for other imported language files.

The Language Manager allows you to define the languages for your project and specify which language will be the reference language.



Creating Languages for a Project

You can create your project languages in the Language Manager. When you create your languages, you can also define the volume offsets for the language files. In some cases, you may need to define volume offsets because your localized assets will consist of dialogue from different studios with different actors and recording conditions, and you may need to balance them and match the levels for each language.

To add a language to a project:

1. From the menu bar, click **Project > Languages**.

The Language Manager opens.

2. From the Available languages list, select the language that you want to add to the Project.



Tip

To select multiple languages, press **Shift** and select the languages you want to include in your project

3. Do one of the following:

Click **Add**.

Double-click the language.

The selected language is moved to the Project languages list.

4. Define the volume offset property (make-up gain) of the audio source for the project language.
5. Click **OK**.

A message box appears warning you that the contents of the Wwise clipboard and the undo history will be cleared.

6. To add the language to your project, click **Yes**.

Related Topics

- [Removing Languages from a Project](#)
- [Defining a Reference Language](#)
- [Importing Language Files](#)
- [Switching to a Different Language Version](#)

Removing Languages from a Project

When you no longer need a language for your project, you can delete it from the project languages list.

To remove a language from your project:

1. From the menu bar, click **Project > Languages**.

The Language Manager opens.

2. From the Project languages list, select the language that you want to remove.
3. Do one of the following:

Click **Remove**.

Double-click the language.

The language is removed from the Project languages list.

4. Click **OK**.

A message box appears warning you that the contents of the Wwise clipboard and the undo history will be cleared.

5. To remove the language from your project, click **Yes**.



Note

Removing a language from the Project languages list will delete all corresponding audio files for that language.

Defining a Reference Language

When you select a reference language for your project, that language will be used in various situations:

- **Importing language files** - When you are importing an audio file, the import conversion settings of the reference file are used.
- **Converting language files** - When you are converting a language file, the conversion settings of the reference language can be selected.
- **Before language files are available** - When certain language sources are not ready, the reference language can be used in their place.

To define the reference language for your project:

1. From the menu bar, click **Project > Languages**.

The Language Manager opens.

2. From the Reference language list, select the reference language for your project.
3. To use the reference language during playback when language sources are not yet available, select the **Use reference language as standin** option.
4. Click **OK**.

A message box appears warning you that the contents of the Wwise clipboard and the undo history will be cleared.

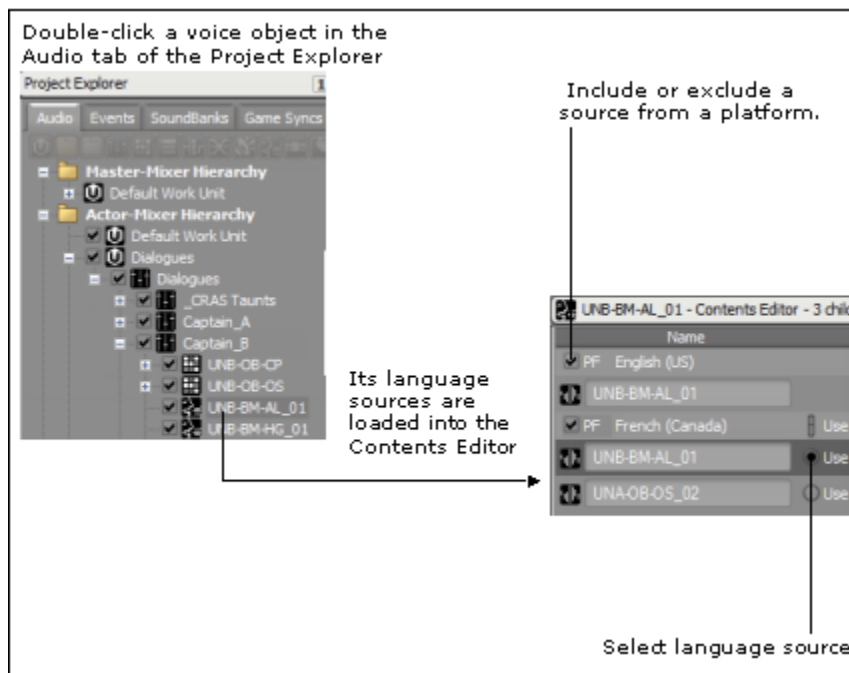
5. To keep the reference language settings for your project, click **Yes**.

Importing Language Files

When your language files are ready, you can import them into your project using the Audio File Importer. When you import these files, language sources are created in the selected voice objects in the Audio tab of the Project Explorer and displayed in the Contents Editor. If you want to localize your project at once, you can import the localized files at the topmost level in the hierarchy.

In the Contents Editor, you have the flexibility of including or excluding the language sources per platform as well as selecting language sources per platform. For more information on how to customize per language and source version as well as deciding which language source to use per platform, refer to:

- [Selecting Sources per Platform.](#)
- [Excluding Project Elements from a Platform.](#)



Note

When you are importing audio files for localizing your project, make sure that the files have exactly the same name as the audio file in the reference language.

To import language audio files into your project:

1. Select an object in the **Actor-Mixer Hierarchy** into which you want to import the language files.



Tip

You can quickly localize your entire project by selecting the top level parent in the **Actor-Mixer Hierarchy** and importing the language files at this level.

2. From the Wwise menu bar, click **Project > Audio File Importer**.

The Audio File Importer opens.

3. Select **Localize languages** as the Import Mode.

The Import as Sound Voice option is selected, and the Sound Voice Options group box and Conversion Settings options become available.

4. From the Conversion Settings list, select the conversion settings that you want to use for these files. Select one of the following:

Apply Default Settings to use Wwise default conversion settings.

Apply Reference Language Settings to use the conversion settings that you have defined for the reference language.

5. From the Destination language list, select the language you are importing.
6. Click **Add**.

The Open dialog box opens.



Note

It is a good practice to add new files to the Originals folder first and then to import them from the Originals folder into your project.

7. Browse to the location of the audio files that you want to import.
8. Select the files, and click **Open**.

The selected files are loaded into the Audio File Importer.

9. Click **Import**.

The Importing dialog box opens where you can view the progress of the file import process.



Note

If there are errors, or conflicts, the Import Conflict Manager opens. For more information on how to deal with these conflicts, refer to [Managing File Import Issues](#).

When the import is successfully completed, the Importing dialog box closes, and you are returned to the Audio File Importer.

10. When you are finished importing audio files, click **Close** to close the Audio Importer.



Note

If you are using source control, you will be prompted to add the imported files to your source control system.

Switching to a Different Language Version

Since your Wwise project contains all the languages that you have defined, you can easily switch from one language to another at any point in the development cycle. The Language Selector list is populated with the languages that you have defined in the Language Manager.

To switch languages:

1. On the toolbar, click the arrow to display the Language Selector list.
2. From the Language Selector list, select the new language.

The localized version of the project will be displayed.

Auditioning and Profiling Language Versions

You can audition, create simulations, as well as profile each language version as you normally would in Wwise. Based on your findings, you can decide to use specific language source versions or sources per platform for the best and most efficient results.

For more information about auditioning, simulating, and profiling with your language versions refer to the following sections:

- [Chapter 40, *Getting to Know the Transport Control*](#)
- [Monitoring and Troubleshooting with the Performance Monitor](#)
- [Chapter 31, *Creating Simulations*](#)

Versions Tips and Best Practices

You may want to review the following sections, which provide you with a series of tips and best practices that can help you better manage your platform and language versions throughout the audio development process.

Converting Multiple Files

You can convert many audio files at once by right-clicking a high-level object in the hierarchy, such as an actor-mixer, and then selecting Convert from the shortcut menu. If you want to set the conversion settings for an entire group of objects, you can right-click the top-level object and select Conversion

Settings from the shortcut menu. You can also multi-select objects in the Project Explorer and then define their conversion settings or convert them all simultaneously.

Strategies for Conversion Settings

Many times, when a studio develops a game, it examines each platform and then allocates a “budget” for such items as the memory footprint, CPU, and bandwidth. The audio team is given a portion of this budget and it is up to them to manage this budget efficiently for each platform. Depending on the platform, these allowances will be different, so the audio designer needs to understand how to efficiently compensate for these differences and still deliver great sound using the various conversion settings available. To see how well you are doing, you can monitor your project's performance in game using the Game Profiler, and adjust the conversion settings as needed to stay within your budget. The following sections look at how different settings can affect the sound designer's budget.

Audio Formats

Audio formats must also be considered in your budget. It is useful to review the characteristics of the different formats supported by Wwise so that you can make good choices for your project. A general rule is that the less a file is compressed, the less processing power is needed; however, an uncompressed file usually uses more memory or disk space.

The following table provides you with some basic suggestions and information to help you decide how to encode your audio files.

File Format	Compression ratio (approx)	CPU	Memory	Examples of how these are commonly used	Limitations
PCM	1:1	Low	High	Sounds requiring high fidelity.	None.
ADPCM	4:1	Moderate	Moderate	Ambient sounds and SFX.	Looping on 64 sample boundaries only.
Vorbis	3-40:1	High	Moderate to very low	Dialogue, music, ambient sounds and SFX.	Slightly larger metadata overhead than other formats, so you should avoid it with very small sounds (less than a few tens of milliseconds). Requires a seek table for seeking.
XMA	2-20:1	Low	Moderate to very low	Ambient sounds and SFX that do not require precise looping.	Large metadata overhead. Streamed sounds can only loop across whole files, and there may be gaps of silence at loop points.

File Format	Compression ratio (approx)	CPU	Memory	Examples of how these are commonly used	Limitations
					In-memory sounds support looping at 128 sample boundaries. Not appropriate for sample-accurate playback.
xWMA	10-30:1	Moderate	Moderate to very low	Dialogue and music.	Can only loop across whole files, and there may be gaps of silence at loop points. Seeking is inaccurate. Not appropriate for sample-accurate playback.
AAC	3-23:1	High (low when using hardware-assisted decoder on iOS)	Moderate to low	Background (not interactive) music.	Very large metadata overhead. Long setup time. Not appropriate for sample-accurate playback.

Using Seek Tables with the Vorbis Encoder

By default, the Vorbis encoder does not use a seek table in order to save disk space. You must, however, use a seek table in the following circumstances:

- When using the Vorbis encoder in combination with the virtual voice "From Elapsed Time" settings.
- When using the Vorbis encoder for interactive music files, especially when the sound engine must start reading the file from somewhere other than the beginning.
- When using Seek actions with the Vorbis encoder.

To enable the seek table for a selection of Vorbis encoded files, do the following:

- In the Conversion Settings dialogue box, click the Edit button for the platform that is using the Vorbis encoder.
- In the Vorbis Encoder Parameters dialogue box, specify a granularity for the seek table by selecting a number from the list.

Smaller seek table granularities require more memory. Sample-accurate seeking (interactive music or seek actions) may incur a CPU spike if granularity is too large, and inefficient I/O usage if it is larger than the disk streaming granularity. As a rule of thumb, pick a seek table size that is smaller or equal to your disk streaming granularity, which is the value used for `AkDeviceSettings::uGranularity` during initialization.

Chapter 31. Creating Simulations

Overview	632
Building a Simulation	633
Managing Playback of Your Simulation	637
Simulating with Game Syncs	640
Fine-Tuning Properties in a Simulation	644
Creating Simulations Tips and Best Practices	648

Overview


At any point in the development process you might find it helpful to build a simulation using the Wwise objects and events you have been working on. To accomplish this, Wwise has created a simulation environment called the Soundcaster where you can play back sound, music, and motion structures asynchronously. This means you can control what plays and when. This can be very handy for testing events, mixing in real time, and so on. The Soundcaster is a powerful tool that can be used for:

- Prototyping and experimenting;
- Developing a proof of concept; and
- Auditioning sounds, music, and motion simultaneously.

Since you can simulate in Wwise alone or by remotely connecting to a game, the Soundcaster provides you with many different uses for your simulation. For any simulation you can choose to:

- Selectively audition the audio or motion for each platform.
- Audition pre-converted audio files.
- Profile your audio and/or motion as it is playing back.
- Mix and test your audio and motion in Wwise by manually simulating the game action.
- Profile your audio and motion in game and in Wwise.
- Experiment with the sounds, music, and motion objects associated with a game object.
- Mix and test your sounds, music, and motion in game.

Before you can create a simulation in the Soundcaster, you need to create a Soundcaster session, which is like a preset that contains the Wwise objects and events used in a simulation. Each Soundcaster layout is saved as a session in the Sessions tab of the Project Explorer, under its corresponding work unit. To help you easily identify a Soundcaster session in the interface, Wwise uses a unique icon to represent it.

Icon	Represents
	Soundcaster Session

You can access the Soundcaster by double-clicking a session in the Project Explorer, or from the menu bar by clicking **Views > Soundcaster**. When you open the Soundcaster from the menu bar, the most recently viewed Soundcaster session is displayed.

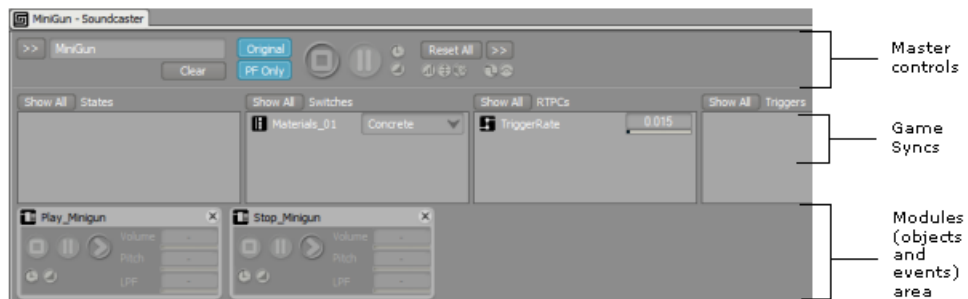


Tip

You can also press **Shift + S** to open the Soundcaster.

The Soundcaster consists of three areas:

- Master controls
- Game syncs
- Objects and events



Using the different areas of the Soundcaster, you can work with its mixing and playback functionalities when you build your simulation.

Building a Simulation

The Soundcaster provides you with as much flexibility as you need to create a simulation for whatever purpose you have in mind. You can add whichever sounds, music, or motion objects you want to work with, and then change your mind as many times as you want. Building a simulation consists of the following steps:

- [Creating a Soundcaster Session](#)
- [Adding Objects or Events to the Soundcaster](#)



Note

You cannot work in the Soundcaster if there are no Soundcaster Sessions in the Sessions tab.

Creating a Soundcaster Session

Before you can use the Soundcaster, you need to create a Soundcaster session. You can create Soundcaster sessions in the following two places in Wwise:

- Sessions tab of the Project Explorer.
- Master control area of the Soundcaster.

To create a Soundcaster Session from the Project Explorer:

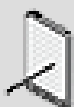
1. In the Project Explorer, switch to the **Sessions** tab.
2. Under the Soundcaster Sessions heading, do one of the following:

Select a work unit or virtual folder and then click the **Soundcaster Session** icon in the Project Explorer toolbar.

Right-click a work unit or virtual folder and click **New Child > Soundcaster Session** from the shortcut menu.

A new Soundcaster session is added to the virtual folder or work unit.

3. Replace the default name with one that best represents the Soundcaster session.
4. Double-click the new Soundcaster session to open the session in the Soundcaster.



Note

You can cut, copy, and paste a Soundcaster session in the Project Explorer using the standard Windows shortcuts. If you no longer need a Soundcaster session, you can delete it by pressing the Delete key.

To create a new Soundcaster session from the Soundcaster:

1. In the Soundcaster, click the Soundcaster Session Selector button (>>) and select **New** from the list.

The New Soundcaster Session dialog box opens.

2. Select the work unit into which you want to create a new Soundcaster session.
3. In the Name field, replace the default name with one that best represents the Soundcaster session.
4. Click **OK**.

The Soundcaster session is created.

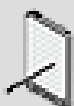
Adding Objects or Events to the Soundcaster

When you add an object or event to the Soundcaster, a module is created. A module represents the object or event together with its properties and behaviors. It also has a series of controls that allow you to modify properties and play back sounds, music, and motion objects.



When you are ready to mix and/or test some of the sounds, music, and/or motion that you have assembled, you can start adding the different modules to the Soundcaster. You can add any of the following to the Soundcaster as modules:

- busses
- Actor-mixers
- Containers
- Sounds
- Motion FX
- Music Containers
- Music Segments
- Events



Note

If a Soundcaster session contains a module that relates to an object or event that has been unloaded from the project, the controls will be unavailable and the title of the module will appear in yellow.

To add an object/event to the Soundcaster:

1. Do one of the following:

Drag an object from the Audio tab of the Project Explorer into the Soundcaster.

Drag an event from the Events tab of the Project Explorer into the Soundcaster.

The blue module insertion indicator appears in the location where you have dragged the object/event. When you release the mouse button, the selected object/event is added to the Soundcaster as a module.

Related Topics

- [Creating a Soundcaster Session](#)
- [Removing Modules from the Soundcaster](#)
- [Re-ordering Soundcaster Modules](#)
- [Playing Back Soundcaster Modules](#)

Re-ordering Soundcaster Modules

When you drag your objects or events into the Soundcaster, you don't need to worry about their order because you can re-order them whenever you want to. Even though you can play back the modules in any order, you may want to arrange them in a particular order to recreate certain sequences or facilitate testing.

To change the order of Soundcaster modules:

1. Drag the module that you want to move to the new location.

The module is moved, and the module formerly in that position is shifted down in the Soundcaster.

Related Topics

- [Creating a Soundcaster Session](#)
- [Removing Modules from the Soundcaster](#)
- [Adding Objects or Events to the Soundcaster](#)
- [Playing Back Soundcaster Modules](#)

Removing Modules from the Soundcaster

If you no longer need a module, or you are ready to create a new simulation, you can remove modules from the Soundcaster.

To remove modules from the Soundcaster:

1. Do one of the following:

To remove one module, click the **Close** button in the title bar of the module.

To remove all modules, click **Clear** in the master control area.

The module is removed from the Soundcaster.

Related Topics

- [Creating a Soundcaster Session](#)
- [Adding Objects or Events to the Soundcaster](#)
- [Re-ordering Soundcaster Modules](#)
- [Playing Back Soundcaster Modules](#)

Managing Playback of Your Simulation

After you have added your modules to the Soundcaster, you are ready to begin using the mixing and playback options.

When you can play back your simulation, you can carry out any of the following tasks:

- Audition platform-specific objects and events.
- Audition pre-converted audio files.
- Audition objects and events associated with specific game objects when connected to the game.
- Play back modules in any order that you want.
- Manage playback for one or all modules at once.
- Apply game syncs.
- Modify properties for modules.
- Modify event actions for objects while playing.
- Audition interactive music.

Specifying Wwise Objects to be Played

When you are creating your simulation, you can be specific in choosing what sounds you want to play based on:

- **Platforms** to include or exclude certain Wwise objects from different platforms in your simulation. For more information about working with sounds and platforms, refer to [Excluding Project Elements from a Platform](#). When you are creating a simulation, you may choose to play only the sounds that are in the current platform, or play all sounds in the module.
- **Converted Sounds** to compare your original versions of audio files with the converted sources. When you convert your imported audio files, Wwise maintains an “original version” of the audio file that you can audition whenever you want. These files have undergone the import conversion process but have not been converted for platforms. By default, the Soundcaster plays the converted sounds; however, you can choose to play the original imported version.

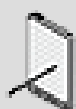
To play back platform-specific sounds and motion FX:

1. In the toolbar, verify that the correct platform for the simulation has been selected.
2. From the Sessions tab of the Project Explorer, double-click the Soundcaster session that you want to define.

The selected session is loaded into the Soundcaster.

3. In the master control area, click **PF Only**.

The PF Only button turns blue and only the objects and events in the current platform can be auditioned.



Note

To play back all objects and events, click **PF Only** again to disable this option.

To play back unconverted sounds:

1. In the master control area, click **Original**.

The button becomes blue, and the imported pre-converted sounds will play in all the modules in the Soundcaster.



Note

To play back the converted sounds, click the **Original** button a second time to deselect it.

To play back Wwise objects associated with specific game objects:

1. From the Game Object list of the Soundcaster module, select the game object whose associated Wwise objects you want to audition.

Only the module objects associated with the selected game object will be played back.

Related Topics

- [Auditioning Sounds, Music, and Motion FX in the Soundcaster](#)
- [Simulating with Game Syncs](#)
- [Fine-Tuning Properties in a Simulation](#)
- [Connecting to a Local/Remote PC or Game Console](#)

Auditioning Sounds, Music, and Motion FX in the Soundcaster

After you have selected what sounds or motion FX you want to play, it is really up to you how you want to audition the modules in the Soundcaster. Using the asynchronous playback functionality, you can audition modules in whatever order you choose, consecutively, overlapping, and so on. Changing the order for playback is a good way to test and experiment as well as help you find the best sequence. You decide how you want to play back the modules based on what you want to do in the simulation. The basic tasks in playback are:

- Play the module.
- Pause the module playback.
- Stop playing the module.

Playing Back Soundcaster Modules

If you want to audition a module, you can use the controls directly in the modules that you want to play. After you have started playing all the modules that you want, in the order that you want, you can pause or stop any modules that you choose. It is not possible, however, to play back an Actor-Mixer or a bus.



You have the option of stopping or pausing all modules using the master controls.



To play back individual Soundcaster modules:

1. In the module, click **Play**.

The associated sound or motion FX will play until finished.

2. If you want to pause the playback, click **Pause**.

The Pause icon turns yellow in the module and in the master control area. To resume play, click **Pause** again.



Note

To resume playing a sequence container from the beginning of the playlist, click **Reset > Reset All Random and Sequence Containers** in the master control area.

3. To stop playback, click **Stop**.



Note

To Pause or Stop all the modules in the Soundcaster, you can use the **Pause** and **Stop** icons in the master control area.

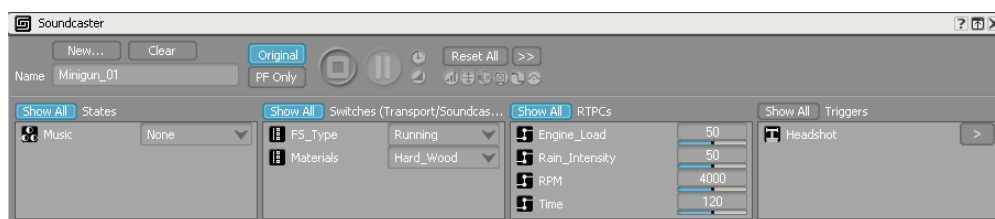
Related Topics

- [Specifying Wwise Objects to be Played](#)
- [Enabling States during Playback](#)
- [Assigning Switches During Playback](#)
- [Simulating Changes in Game Parameters](#)
- [Real-Time Mixing and Positioning](#)
- [Using the Reset Function](#)

Simulating with Game Syncs

In the Soundcaster you also have access to the game syncs that you have created for your project. You can audition the states, switches, RTPCs, and triggers as they are applied to your sounds, music, and motion FX objects, both in game and in Wwise. By connecting to your game and adding the appropriate modules to the Soundcaster, you can audition, test, and mix, using game syncs, in real time with the game action.

When the Show All buttons are selected, all created game syncs are displayed in the Game Syncs area of the Soundcaster. Otherwise, only the game syncs associated with the Wwise objects loaded in the Soundcaster will be displayed.



Enabling States during Playback

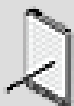
After you have added your modules to the Soundcaster session, and determined which Wwise objects will play back, you can enable states during playback. When you drag an object into the Soundcaster, the state groups and states enabled for the Wwise objects in the module are added to the States list. If there are no states displayed, click **Show All**. All state groups and states will then be displayed in the States area. To learn more about creating states and assigning objects to states, refer to the following sections:

- [Working with States](#)
- [Assigning States to Objects and Busses](#)

To enable a state during playback:

1. In the states list, select the state that you want to apply.

The state will be applied to all Wwise objects in the Soundcaster that subscribe to the state.



Note

To return to the default state specified for the Switch container, click the Selector (>>) button in the master control area, and select Reset All States from the menu.

Related Topics

- [Assigning Switches During Playback](#)
- [Simulating Changes in Game Parameters](#)
- [Calling Triggers During a Simulation](#)

Assigning Switches During Playback

After you have added your modules to the Soundcaster Session and determined which Wwise objects that you want to audition, you can enable switches to be applied during playback. When you drag an object into the Soundcaster, the switch groups and switches applied to the Wwise objects in the module are added to the Switches area. If there are no switches displayed, click **Show All**. All switch groups and switches will then be displayed in the Switches area. To learn more about creating switches and how switches are used, refer to the following sections:

- [Working with Switches](#)
- [Defining the Type of Switch Container](#)

To assign a switch to your modules:

1. From the switches list, select the switch that you want to apply.

The switch containers that have subscribed to the selected switch group will play the Wwise objects that correspond with the switch that you have chosen.



Note

To return to the default switch specified for the switch container, click the Selector (>>) button in the master control area, and select Reset All Switches from the menu.

Related Topics

- [Enabling States during Playback](#)
- [Simulating Changes in Game Parameters](#)
- [Calling Triggers During a Simulation](#)

Simulating Changes in Game Parameters

After you have added your modules to the Soundcaster and determined which Wwise objects that you want to audition, you can test the property values that you mapped to the game parameters. To learn more about creating game parameters and mapping property values to them, refer to the following sections:

- [Managing Game Parameters Used in RTPCs](#)
- [Assigning Wwise Properties to Game Parameters](#)

When you drag an object into the Soundcaster, the associated game parameters and a corresponding property slider are added in the RTPCs area. If there are no game parameters displayed, click **Show All**. All game parameters that you have created will be displayed in the RTPCs area. The slider represents the range of game parameter values. Since you have already mapped these values to Wwise property values, when you change the parameter values, you automatically change the property values.

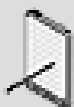


You can audition these property changes during playback in your simulation.

To modify game parameter values during playback:

1. During playback, use the RTPC slider to change the game parameters values.

The properties for the associated object will change based on the mapping you created between the game parameter and the object properties.



Note

To return the game parameters to their original settings, click the Selector (>>) button in the master control area, and select Reset All Game Parameters from the menu.

Related Topics

- [Assigning Switches During Playback](#)
- [Enabling States during Playback](#)
- [Calling Triggers During a Simulation](#)

Calling Triggers During a Simulation

After you have added your modules to the Soundcaster and determined which Wwise objects that you want to audition, you can also test the triggers that will launch musical phrases called stingers over your music. In this way you can simulate what is happening at key points in the game when a trigger calls a stinger to play over the current music. To learn more about creating triggers and creating the stingers for them that you will be auditioning in the Soundcaster, refer to the following sections:

- [Working with Triggers](#)
- [Chapter 28, Using Stingers](#)

When you drag an object into the Soundcaster, the associated triggers are added in the Triggers area. If there are no triggers displayed, click **Show All**. All triggers that you have created will be displayed in the Triggers area.

To call a trigger during playback:

1. While the music object in the module is playing, select the trigger that you want to audition from the Triggers list.



2. Click the **Call Trigger** icon.

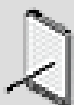
The corresponding stinger will play over the currently playing music object. You can continue to select triggers and play back the corresponding stingers to simulate the music in the game.

Related Topics

- [Auditioning Sounds, Music, and Motion FX in the Soundcaster](#)
- [Working with Triggers](#)
- [Chapter 28, *Using Stingers*](#)

Fine-Tuning Properties in a Simulation

As you are auditioning the events and objects in the Soundcaster, you might want to try out different property values and mixing. You can modify the properties for each sound, music, and motion FX objects directly in the Soundcaster.

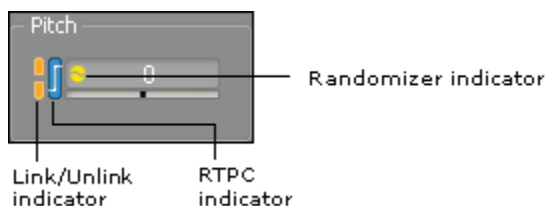


Note

If you are working with events, you need to open the Event Manager to make any changes to the property actions. For more information about working with properties and events, refer [Adding Actions to an Event](#).

Property Indicators








You may notice that certain property values in the modules have one or more indicators beside them. These indicators show whether the property value is linked to other platforms, whether the property value is associated with a game parameter using RTPCs, and whether a Randomizer has been applied on the property value. For more information about how these indicators are used in Wwise, refer to [Property Indicators](#).



For more information on linking/unlinking property values, using RTPCs, and randomizing property values, refer to the following sections:

- [Customizing Object Properties per Platform](#)
- [Controlling Property Values Using Game Parameters](#)
- [Enhancing Sound and Motion by Randomizing Property Values](#)

In addition to these property indicators, the Soundcaster also contains playback indicators that turn blue when particular behaviors or actions occur during playback. These are visible in the master control area, and in the individual modules. The following table lists the additional property and action parameter indicators in the Soundcaster.

Icon	Name	Indicates
	Delay	A delay has been applied to an event or a random or sequence container.
	Fade	A fade has been applied to an event or a random or a sequence container.
	Set Volume	A set volume action has been applied to a sound, music or motion FX object in an event.
	Set Pitch	A set pitch action has been applied to a sound or motion FX object in an event.
	Mute	A mute action has been applied to a sound, music, or motion FX object in an event.
	Set Low Pass Filter	A set Low Pass Filter action has been applied to a sound or music object in an event.
	Enable Bypass	An Enable Bypass action has been applied to a sound, music, or motion object in an event.

For more information on adding these properties to objects, refer to the following sections:

- [Playing All Objects Within the Container](#)
- [Setting Properties for an Event Action](#)

Real-Time Mixing and Positioning

In your simulation you can use the property controls to do the following before and during playback:

- **Mix in real time** to modify the volume, pitch, low pass property values. For more information about mixing these properties, refer to [Defining Relative Properties \(Volume, Pitch, LPF, HPF\)](#).
- **Modify positioning** to modify the 2D or 3D sound or motion propagation properties. For more information about positioning, refer to [Chapter 10, Defining Positioning for Sound and Motion](#).





Tip

To modify the properties directly in the Property or Event Editor, double-click the title bar of the module.

Using the Reset Function

After auditioning an event module in the Soundcaster, you may decide that you want to make changes to the properties of the objects associated with the event. Event actions can temporarily change the properties of objects. Before changing the properties, you should reset the properties back to their original values. For more information on working with events and event actions, refer to [Working with Events](#). You can reset event actions in each module, and in the master control area of the Soundcaster for all modules.



Note

In the master control area, you can also clear all music tracks from being forced to play in the Soundcaster.

To reset specific event actions in a Soundcaster module:

1. In the module, click and hold the **Reset** button.

The Reset menu opens.

2. In the Reset menu, select one of the following:

Reset All to reset all objects to their original settings.

Reset Mute to clear the mute actions that have been triggered for the objects.

Reset Pitch to clear the pitch actions that have been triggered for the objects.

Reset Volume to clear the volume actions that have been triggered for the objects.

Reset Low Pass Filter to clear the low pass filter actions that have been triggered for the objects.

Reset Bypass Effect to clear all bypass actions that have been triggered for the objects.

To reset event actions in the master control area:

1. In the master control area, click the Selector button (>>).

The Reset menu opens.

2. In the Reset menu, select one of the following:

Reset All to reset all objects to their original settings.

Reset All Set Mute to clear all mute actions that have been triggered for the objects.

Reset All Set Pitch to clear all pitch actions that have been triggered for the objects.

Reset Set Volume to clear volume actions that have been triggered for the objects.

Reset All Set Low Pass Filter to clear all Low Pass Filter actions that have been triggered for the objects.

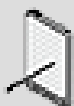
Reset All Bypass Effect to clear all bypass effects actions that have been triggered for the objects.

Reset All States to clear all Set States actions for objects.

Reset All Switches to clear all Set Switch actions that have been set for the objects.

Reset All Music Tracks Force Usage to clear forcing the playback of a specific track in the Soundcaster.

Reset Position to reset the position of the listener within the Attenuation Preview control to its default position.



Note

The Reset function in the master control area includes additional commands. For information about resetting random and sequence containers, refer to [Playing Back Soundcaster Modules](#). To reset states, switches, and game parameters, refer to [Simulating with Game Syncs](#).

Related Topics

- [Simulating with Game Syncs](#)
- [Managing Playback of Your Simulation](#)

Creating Simulations Tips and Best Practices

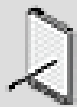
You have a great deal of flexibility in creating your simulations in Wwise and you can use them effectively to profile, create proof of concept and to test your ongoing work in Wwise or in game. Here are some concepts to consider to help you achieve the best results for your game.

Real-time Mixing and Object Properties

When you are connected to a game or the Game Simulator, you can modify the values of the following relative properties from within Wwise in real time:

- Volume
- Pitch
- Low pass filter

To be able to do this, you need to load the object whose property you want to modify in the Transport Control or the Soundcaster. If the object is not loaded, your changes will not take effect because the object is not registered in the sound engine. For the Transport Control, certain objects such as actor-mixers cannot be loaded, but you can load a child object of the actor-mixer and this will register its parent objects in the sound engine. Since you can load an actor-mixer in the Soundcaster, this is not a problem. After you have registered the object, it will remain registered for the time that you are connected to the game.



Note

Keep in mind that if you pin an object in the Transport Control, other objects cannot be loaded until you unpin the first object. If, however, you have loaded the object in the Soundcaster, the object will be registered in the sound engine.

For some properties, including randomizers, and effect, attenuation and source plug-in properties, the changes that you make will not be reflected until the next time that object plays in-game.

Chapter 32. Managing Memory in Wwise

Overview	650
Understanding the Components of the Memory Manager	650
Setting the Size of Your Memory Pools	651
Troubleshooting Memory Problems	654
Optimizing Memory Pools	654
Memory Management Tips and Best Practices	660

Overview

With Wwise, the possibilities to create rich and sophisticated soundscapes for your game are endless. It is important, however, to remember that memory resources are limited on the various platforms. When designing the audio for your game, you should keep these limitations in mind. As you build the audio elements within your project, you can use the tools provided by Wwise, such as playback limits, virtual voices, and the volume threshold, to efficiently manage the memory in your game.

The Profiler provides you with a wealth of information that can help you troubleshoot memory issues in your game. To fully understand the information provided in the Profiler and to take full advantage of the memory saving tools in Wwise, you must first understand how Wwise manages all the different aspects of your project, including the final media files, sound structures, events, game syncs, effects, and so on.

This chapter provides you with a general overview of the different components of the memory manager as well as some tips and best practices on how best to manage the memory for your game. For more information about the different components of the Memory Manager, refer to [the Wwise SDK documentation](#).

Understanding the Components of the Memory Manager

The Wwise sound engine has a Memory Manager to manage the memory dedicated to the audio in your game. The Memory Manager is initialized by the audio programmer at the startup of your game. When initializing the Memory Manager, the programmer must also create a specific number of memory pools to manage the different audio components in your game.

There are three different types of memory pools in Wwise:

- **Default memory pool** - a general usage pool for high-level sound structures. This pool mostly contains the following:
 - Audio structural content.
 - Command queue for posted events.
 - Allocations for registered game objects, game object positions, listeners, RTPCs, switches, states, and so on.
- **Lower engine memory pool** - The main audio processing pipeline pool. This pool is generally used for audio playback, audio processing, memory allocated for effects, and so on.
- **Media memory pool** - A general usage pool that contains a copy of the media files contained on the disk.

When creating the memory pools for your game, the audio programmer must also define their size. Once a pool is created, it cannot be resized. Memory pools can be created and destroyed during the entire lifetime of the Memory Manager. Pools are segmented into blocks of equal size representing the smallest allocation unit.

You can override the Memory Manager with any other technology, from internal to Unreal, example. For more information on implementing a custom memory manager, refer to the section [Overriding the Memory Manager](#) in the Wwise SDK documentation.

How Memory Pools Are Used in Your Game

At a very basic level, when a SoundBank is loaded into memory, the sound engine performs two functions, it:

- Parses the SoundBank's metadata and stores the sound hierarchy and event information within the default memory pool.
- Copies any media files to a user-defined memory pool.

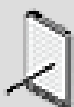
When an event triggers audio in your game, the lower engine memory pool performs any necessary audio processing before playing back the final audio.

Setting the Size of Your Memory Pools

The default size of the internal memory pools of the sound engine are very large only to ensure that when you try the sound engine for the first time, it has enough resources to run your worst-case scenario.

The default size for each pool is actually:

- 16 MB for the default memory pool
- 16 MB for the lower engine memory pool



Note

If the sample code is used to integrate the streaming manager, 8 MB are also allocated for the streaming buffers.

The size of the default memory pool will ultimately depend on the type of game you are creating. For example, if your game has gigabytes of audio assets, but rarely plays more than four or five sounds at a time, with no or few effects, then the memory requirements of the default memory pool may be larger, but the requirements of the lower engine memory pool may be smaller. If, on the other hand, your game has few sounds, but plays them all simultaneously with lots of reverb, echo, and other environmental effects, like in a first-person shooter, then the memory requirements of the default memory pool may be smaller, but the requirements of the lower engine memory pool may be larger.

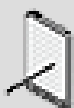
The size of the lower engine memory pool will ultimately depend on what you plan to do with the sound engine. If your game will play a lot of voices simultaneously (including virtual voices), and will be processing a large number of effects, then you will need more memory in this pool.

On the PC, with no banks loaded or sounds playing, the amount of memory used should be around 150kB. You can see the exact amount in Wwise, by switching to the Profiler layout and looking at the Total Used Memory value in the Performance Monitor. This value will obviously go up with the number of assets loaded, voices playing, game object positions, and so on.

The required size of your memory pools will depend on a number of different factors, including the quality of your audio, the number of sounds loaded, the number of voices playing simultaneously, the complexity of your sound structures, the encoding method chosen, the type and number of effects that require processing, the number of 3D positioned sounds, switches, and so on.

The only reliable way to know how much memory is required is to do the following:

- Assign an arbitrary amount of memory to the pools. If you can, start by using the default memory sizes.
- Connect Wwise to your game.
- Run a busy game scenario with the Profile version of the sound engine.
- Look at the Memory tab of the Advanced Profiler. The Peak Used column should give you a good indication of what the pool sizes should be. Typically, the optimal size for each pool is 15-20% above the peak memory usage value.
- Subtract the size of the Communications, Monitor, and Monitor Queue memory pools. These pools contain debug-only allocations which are not created in the Release version. Subtracting the size of these pools should give you a good idea of the release-version memory usage.



Note

You can also profile sound engine memory usage without connecting to a game. Simply start a capture session and then play one or more events and/or objects in the Transport Control or Soundcaster. The amount of memory used to play these events and/or objects will be displayed in the Memory tab of the Advanced Profiler.

Choosing Worst Case Scenarios for Testing

Each memory pool may have a different worst case scenario. For example, playing sixty-four sounds at once greatly increases usage of the lower engine

pool while only slightly increasing usage of the upper or default memory pool. On the other hand, loading twenty banks all at once increases upper engine memory usage without affecting lower memory pool usage. Therefore, we recommend that you don't use just one worst case scenario for all your tests, because you are likely to have many diverse situations in your game that will most likely use memory differently.

One particular situation to test occurs when SoundBanks are loaded and unloaded between levels, because this causes a peak in memory usage in the default memory pool. For example, when switching between Level 1 and Level 2 in your game, you may experience a peak in memory usage because there may be a brief period when both SoundBanks are loaded in memory.



Caution

When performing tests to check peak memory usage for the Windows platform, you must ensure that the speaker setup configuration of the PC running the game is set to 5.1 or better. This is necessary because the sound engine performs some optimizations when the speaker setup is stereo, and therefore may use less memory than in 5.1.

What Size Should I Use to Get Started?

In most cases, it is best to start off using the default memory pool sizes (16 MB per pool). By having such large pool sizes, you can easily find out what your real memory usage is and then you can trim down the pools accordingly. If you are more restricted to the amount of memory you can assign to these pools (16 MB is too much), you can start by using the following values:

- Default memory pool: 2MB
- Lower engine memory pool: 2MB
- Streaming management: 2MB

The streaming buffer size will depend on if you plan to stream a lot of sounds, the quality of the sounds, and the safety timeline of the streaming sounds. The default value of 8MB would generally be seen as a waste of memory for most games, but again it is best to start out large and trim down as your memory usage becomes clearer.

Memory Pool Sizes and Platforms

Memory pool sizes are entirely platform independent. When setting the pool size for each platform, it is important to understand that the size of a memory pool depends much more on audio content and usage than it does on the platform. This means that you can use the same pool sizes to start with and then

tailor them to the particular platform using the memory usage readings in the Profiler. The exception to this would be the lower engine memory pool on the Wii, which requires much less memory than the other platforms due to the fact that it has no software mixing buffers. In this case, a value of 1MB for the lower engine memory pool would be a good starting point.

Troubleshooting Memory Problems

After setting the initial size of your memory pools, you can run the game to see if you run into any problems. The Profiler will be a valuable tool at this stage to detect any problems that may arise.

When you use the Wwise Profiler, a warning notification is sent to the capture log for every allocation failure that occurs in the game. You can check this list of notifications to find out which memory pools were lacking in memory at each point during gameplay.

For more information about using the Profiler, refer to [Chapter 33, Profiling](#).

Running Out of Memory

The sound engine reacts differently to 'out of memory' conditions depending on when the condition occurs. The following table describes what happens when memory is missing during a series of different scenarios:

Scenario	Result When Memory Missing
Initializing the sound engine	The initialization fails.
Loading a SoundBank	The SoundBank load fails.
Starting a transition for a property, such as volume	The transition is skipped and the parameter jumps directly to the target value without transition.
Playing a sound	This depends on whether the Memory Threshold has been enabled in the initialization of the Sound Engine. By default, the Memory Threshold is disabled. In this case, the playback will simply fail. If the Memory Threshold is enabled, then the priority rules will apply. It is highly recommended that you enable the Memory Threshold feature.

Optimizing Memory Pools

In order to optimize your memory pools, you need to understand where you can save on memory. The following items can cause high memory usage:

- Loading banks increases memory usage in the default memory pool. Note that each bank uses a different amount of memory. Memory used by a bank in the Default memory pool does not depend on the physical size of the bank but on the number of sounds and events that it contains.
- Some effects, including reverb and delay, consume a certain amount of memory in the lower engine pool when playing.

- Playing multiple sounds at once drastically increases the amount of memory used in the lower engine pool.
- Sending multiple actions in a short amount of time increases the memory usage of the default memory pool.
- Registering game objects, setting 'per object' parameters, and setting object positions all use a small amount of memory in the default memory pool. Be aware, however, that unused game objects must be unregistered to free up this memory. Otherwise, the amount of memory being used constantly increases.

When optimizing your memory, you need to think in terms of the following memory pools:

- [Lower Engine Memory Pool](#)
- [Default Memory Pool](#)
- [Media Memory Pools \(SoundBanks\)](#)

Lower Engine Memory Pool

The memory in this pool is used to play sounds. It is directly influenced by the number of sounds playing at the same time. It is also influenced by the number and type of effects that are used at the same time. So to trim this down, you need to ask yourself how many sounds do you want to hear at the same time. Some games will rarely have a scenario where more than 10 sounds are heard, others will have 100. When setting the size of your memory pools, remember to use your worst case scenario.

As a guideline, we have profiled a few games using the Xbox360 and have come up with the following guidelines for the lower engine memory pool:

- 1 MB will let you play between 50 and 80 voices.
- 2 MB will let you play between 130 and 170 voices.

As you can see, it scales mostly linearly but really depends on which codec is used, how many effects are used, among other things.

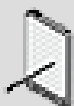
You also need to ask yourself if playing back 170 sounds simultaneously actually adds any value to your game. At a certain point, the human ear can't decipher the different sounds and it just becomes noise. You need to find the appropriate number of voices that is valuable for your game. This, of course, will require some experimentation. You can profile multiple scenarios in your game and use the Memory tab of the Advanced Profiler to note how much of the resources are used.

To reduce the memory used in the lower engine pool, you need to reduce the number of simultaneous voices. This can be done by using any of the following strategies:

- **Playback Limits** (Advanced Settings). For example, do you really need to hear 50 bullet ricochets? Most likely no, so you might want to limit those sounds to a more reasonable number, such as 15. Note that you can set limits on busses as well.
- **Playback Priority** (Advanced Settings). For example, bullets could be less important than dialog. This means that bullets would get kicked first if there are too many sounds. Use in conjunction with Playback Limits.
- **Distance-based Priority Offset** (Advanced Settings). Objects that are far are usually less important than closer ones. For example, we don't need to hear bullets that are 10 meters away if there are 15 other bullet sounds closer than that.
- **Memory Threshold** (SDK). When initializing the sound engine memory pools, you can specify a percentage of the size of the pool at which voices will start to get killed based on priority. This will put a hard limit on memory that will avoid almost all Out Of Memory conditions in a controlled way. Note that memory can never be completely used because of fragmentation, so a good starting value would be 0.9 or 90%.
- **Volume Threshold** (Project Settings). This will help kill the sounds that are too faint to be heard. This goes hand in hand with the Below Threshold Behavior and also the Attenuation settings, where farther usually means fainter. The default value is -80.
- **Below Volume Threshold Behavior** (Advanced Settings). The least expensive option (CPU and memory) is 'Kill Voice'. This option is useful for short sounds or looping ambience sounds, where it kills the voice, but not the loop. The second preferable option is 'Send To Virtual' + 'Play from beginning', then 'Send To Virtual' + 'Resume' and then 'Send To Virtual' + 'Play from elapsed time'. 'Continue to play' and 'Play from elapsed time' are the most costly options. Be aware that Wwise's default value is 'Continue To Play'.

Default Memory Pool

The Default memory pool contains the structural metadata related to the sounds and events loaded into memory. It contains all the properties of the objects in your project necessary to implement the behaviors defined in Wwise. It also contains all the registered game objects and their related information, including game sync values, position, orientation, and so on. As more banks are loaded into memory, more metadata is added as well, requiring a larger default memory pool size. The size will ultimately depend only on the amount of sounds that can possibly be played in one scenario, level, map, game area, and so on.



Note

The default memory pool does NOT contain any media.

Each sound structure and event combination takes on average about 300 bytes of memory in the default memory pool. If, for example, your project had 50,000+ sounds (including dialog), this would require approximately 15MB of memory for the structural data only. As you can see, you can't load it all into memory at once, so you need to load and unload structural data in the same way as you load and unload media.

Based on the Wwise projects we have seen to date, sound structures account for the biggest part of the default memory pool (anywhere between 25-50%) due to the sheer number of them loaded into memory. Of course, the less sounds you have loaded, the smaller the default memory pool will be. The next biggest user of memory is the event metadata, which usually takes about 10% of the pool. Events are very small, so even though you may have many, they don't take up that much space. The Random/Sequence container structure is probably the biggest single user of memory taking about three times the memory of a simple sound. However, since there are generally far fewer random/sequence containers than either sounds or events, these take up much less of the pool's memory as a whole.

As a general rule, open-world games tend to have a larger memory requirement because more sounds need to be ready to play. Such games will need anywhere from 5 to 8 MB for the default memory pool. Other types of games can get by with a much lower usage, for example 2-3 MB. Please note that these numbers have been provided to give you an idea of what other games using Wwise have required, but in no way guarantee that your particular game design will fit within these memory ranges.

The following best practices can help you reduce the memory used by the default memory pool:

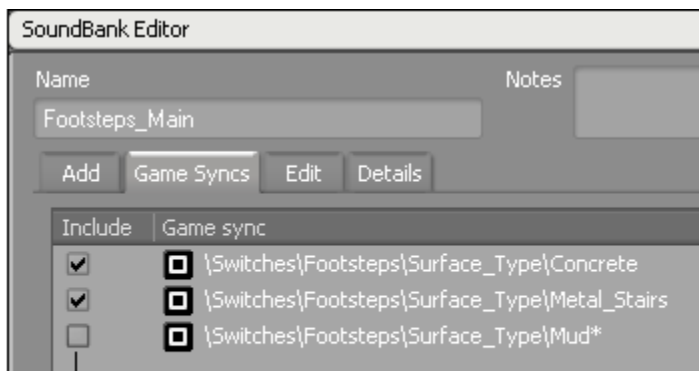
- Split big banks with lots of sound structures and events into smaller banks. Load and unload banks dynamically, as needed. Make sure that your banks are not only organized per-character, but have more situational divisions.
- Reduce the number of events by using the `ExecuteActionOnEvent` API. Play/Stop pairs can be replaced by a single Play event and a call to `ExecuteActionOnEvent` for the Stop. The same is true for Pause/Resume, for the same Play event.
- Tightly manage your game objects. Unregister them as soon as their role is finished. Avoid keeping a pool of unused game objects alive; there is absolutely no gain in doing this, but it can cost in memory. For example: when an NPC dies, you should unregister its game object. Don't re-use it for something else. Register a brand new game object when one is needed. As a general rule, if you have thousands of game objects alive, it is too many.
- Use virtual folders to organize sounds and not Actor-Mixers. Virtual folders don't take memory, actor-mixers do. Only use actor-mixers if the objects actually share similar properties other than the default. In this case, you save

memory using the actor-mixer because the property is there once only. Of course, this also depends if the actor-mixer is referenced by events, such as SetVolume or SetPitch.

- Try to reduce the size and complexity of large hierarchies. A common example of a large hierarchy would be an Impact hierarchy or a Footstep hierarchy. With lots of variables, it can grow large and unfortunately can't be split (the media can be, but not the structure data). Here are a few ways to reduce such a hierarchy:
 - Use RTPCs if the only thing changing is a simple property (same samples but different volume/pitch/randomizer and so on.).
 - Replace some levels (or all) of the switch hierarchy with a Dialogue Event with equivalent State Groups. This is useful for variables that change often. For example, in an impact hierarchy, there is no pre-determined surface that will be hit, you only know when the event occurred. Instead of using a switch container, which needs to keep the switch value, you could use a Dialogue Event with a state group with the same possible state values. One of the advantages is that the Dialogue Event can easily re-use the same samples or sub-structures. Also, the Dialogue Event uses no memory per-game object. Also remember that you can link a path to a switch container, adding another level of variables.
 - An example where this can be used is for footsteps. The Dialogue Event would have maybe one or two state groups, such as Step Type (Walk/Run) and Surface Type, and each path could link to a switch container for 'Footwear Types' with values, such as 'Boots', 'Civilian shoes', 'Barefoot'. The logic being that the 'Footwear' doesn't change often in the life of the game object but the other two variables do.
 - Another advantage to this method is that dialogue events can live in an event-only bank without the associated sound structures that it plays. This means that you can split the complex sound structures associated with the event between many banks. Using the footsteps example, this would mean that you can have the footstep Dialogue Event in BankEvent, have the Concrete-related structure and media in BankConcrete, the Dirt-related structure and media in BankDirt, and so on. For this to work, you would simply have to load and unload SoundBanks dynamically as the surface types are about to be encountered.
 - Split your switch container hierarchy into multiple SoundBanks. When you add a switch container to a SoundBank, all of its sub-branches are also included automatically. You can, however, exclude some of these sub-branches manually on either the Game Syncs or Edit tabs of the SoundBank Editor. Let's say, for example, that you have a Footstep hierarchy, where the top-level switch variable is the surface type. If not all surface types are required at all times in your game, you could split the switch container hierarchy across different banks and then load only those surface types that are necessary, depending on the context in game. For

example, you could have a main Footstep bank containing the surfaces that are encountered everywhere in the game, such as Concrete and Metal Stairs, and then have other contextual banks for specific surfaces that are only used in one scene or section of the game, such as Mud, for example.

Replace impact-type wave files used to create variations of a sound with a SoundSeed Impact equivalent. Replacing 10 different 'clang' sounds with one is not only economical, but it will give you more than 10 possible variations. Don't neglect this possibility, as it can really make a difference.



The structural metadata related to the 'Mud' surface is removed from the Main bank and stored within a separate bank.

Media Memory Pools (SoundBanks)

The amount of memory taken by SoundBanks is mostly dictated by the sound data in it. Controlling the amount of memory used by your media can be done by using the following techniques:

- Split big banks with lots of sounds into smaller banks. Load and unload the banks dynamically, as needed.
- Stream more sounds from disk (Sound object properties).
- Use the PrepareEvent/UnprepareEvent API.
- Compress the audio more (Conversion Settings, codec, and so on).
- Use a lower sampling rate. Also look at the Automatic Sample Rate Detection feature (Conversion Settings).
- Replace impact-type sounds with a SoundSeed Impact plug-in equivalent. You can save a great deal of memory by replacing any random containers you may have created with multiple sound variations in them. Note that SoundSeed Impact can be used to create variations of some non-impact sounds as well.
- Replacing wind-type sounds with a SoundSeed Wind/Woosh plug-in equivalent. Wind ambiances tend to be long looping sounds, which can take a great deal of media space. The SoundSeed Air plug-ins can be used to create a variety of different sounds, including blade wooshes, propellers, wind

rushing through a car with open windows, ventilation noises, or any noisy sound, such as ocean waves or the sound of a highway in the distance.

Memory Management Tips and Best Practices

The following sections provide you with a series of tips and best practices that can help you effectively manage the audio memory in your game.

Using the Memory Threshold

As mentioned briefly in an earlier section of this document, you also have the option to define a memory threshold for one or more of the sound engine's memory pools. The audio programmer can define a threshold for a memory pool by modifying the following values in the initialization parameters of the sound engine:

- `AkInitSettings::fDefaultPoolRatioThreshold`
- `AkPlatformInitSettings::fLEngineDefaultPoolRatioThreshold`

By default, the memory threshold is disabled using a default value of 1 (or 100%). You can enable it by setting a value ranging anywhere between 0 and 1.

When the memory threshold is disabled, the memory allocator will work normally. When enabled, the engine periodically checks that the percentage of memory being used is below the specified threshold. If it goes over the threshold, the system will start dropping sounds with the lowest priority.

In situations where the memory threshold is not used, the engine will honor priorities of sounds, but in low-memory conditions, the engine may not have enough memory to play a sound with a high priority, which means that it would get dropped. When the memory threshold is used, sounds with lower priorities will be dropped to free up space for the higher priority sound.

Dealing with Memory Fragmentation in the Sound Engine

Whenever you are dealing with chunks of data being added and removed from memory, you will have some amount of fragmentation. Wwise does not, by itself, deal with memory fragmentation, but the memory pools have been designed to reduce the amount of fragmentation that occurs.

The default memory pool, which is used mainly for sound structure metadata, consists mainly of small objects, so fragmentation is not really an issue. The lower engine memory pool, which is reserved for the audio pipeline and processing, does consist of larger blocks, but these blocks all get freed when the audio dies down. For both of these memory pools, by simply allocating an overhead of approximately 10% to these pools, you can avoid fragmentation almost entirely.

As for the audio data that is stored in banks, it is copied in a separate user-defined pool for each `LoadBank()` call, giving the audio programmer great control over the memory management of audio data. To minimize any memory fragmentation that may occur, the audio programmer can try using fixed-block-size pools by passing the `AkFixedSizeBlocksMode` flag to `CreatePool`, with the block size being your largest bank for each pool. This will allocate every bank as if it were the size of the largest bank. Of course, you can have more than one pool, where each pool gives you a given number of slots/blocks for a certain size of bank. Memory blocks of `AkFixedSizeBlocksMode` pools are obtained through `GetBlock()` and always have the same size. By using this technique, fragmentation can be completely avoided.

Chapter 33. Profiling

Overview	663
Understanding the Different Types of Profiling in Wwise	664
Connecting to a Local/Remote PC or Game Console	669
Capturing Data from the Sound Engine	672
Monitoring and Troubleshooting with the Performance Monitor	683
Keeping Track of Objects and Listeners with the Game Object Explorer.....	686
Examining Objects with the Game Object 3D Viewer	691
Evaluating Game Syncs with the Game Sync Monitor	696
Profiling Tips and Best Practices	697

Overview

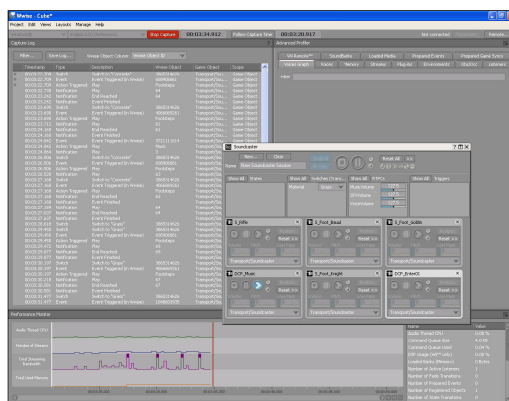
One of the biggest challenges for game developers is to create an extraordinary gaming experience while respecting the limitations of the various platforms. In Wwise, there are many ways to tailor your game audio and motion to the various platforms. You can, however, take it one step further by using the Game Profiler to test how your audio and motion performs on each platform.

The Game Profiler allows you to profile specific aspects of your game at any point in the production process on any platform. You can profile locally in Wwise or you can profile by connecting to a remote game console. Both methods capture profiling information directly from the sound engine. By monitoring the activities of the sound engine, you can detect and troubleshoot specific problems related to memory, voices, streaming, effects, and so on. You can also use the Game Simulator and Soundcaster to profile prototypes even before they have been integrated into your game.

Profiling Locally

Before your audio and motion are integrated into your game, you can profile any aspect of your Wwise project by triggering events, sound and motion objects, and game sync changes in the Soundcaster.

1. Switch to Profiler layout.
 2. Populate Soundcaster with events, objects, and busses.
 3. Start capture.
 4. Trigger events, objects, or game sync changes in the Soundcaster.
- Information from the sound engine is captured and displayed in the different views of the Profiler.



Profiling in Game

After your audio and motion are integrated into the game, you can connect Wwise to the game and then profile in real time as the game is being played. Once connected to the game, you can also tweak and mix the relative properties of your objects in real time in game.



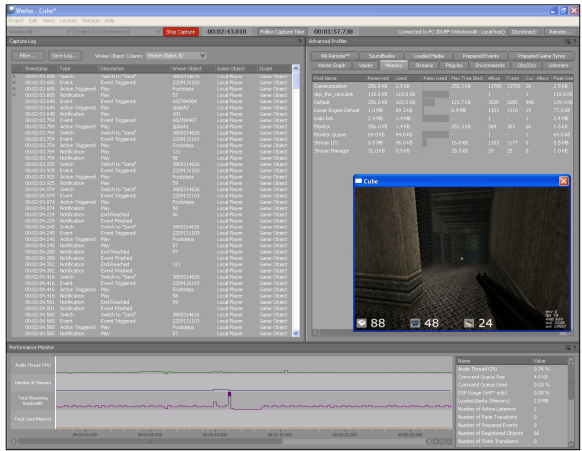
Caution

When profiling, it is recommended that you connect to the Profile build configuration of the Wwise sound engine, even

from the Debug build of your game, because the performance of the Debug configuration has not been optimized.

1. Start your game.
2. In Wwise, switch to the Profiler layout.
3. Use the Remote Connection feature to connect to the game.
4. Start capture.
5. Start playing your game.

Information from the sound engine is captured and displayed in the different views of the Profiler.



Understanding the Different Types of Profiling in Wwise

In Wwise you can perform two types of profiling:

- Game profiling
- Game object profiling.

Game profiling focuses on performance requirements and demands from the point of view of the sound engine and the various components that make up your project. It demonstrates in real time the cumulative effect the sound and motion in your project has on platform performance, and allows you to examine the impact of individual voices.

Game object profiling also analyzes the output of the sound engine, but from the point of view of individual game objects. Game object profiling tracks game objects so that you can observe their movements and behavior in real time. In this way, you can find out if certain game objects are problematic.

Game objects are discrete entities that exist within a game. They are registered or created by the audio programmer for all objects or elements within your game that can emit a sound, including player characters, non-player characters (NPCs), weapons, vehicles, monsters, ambient objects, such as torches, and so on. Game objects, which may be programmed to move independently, can have sounds, music, or motion FX associated with them. The game object profiling tools (the Game Object Explorer, 3D Game Object Viewer, and Game Sync Monitor) work together to examine game objects in a game or simulation.

Profiling the Sound in Your Game - Example

Let's say you are making a fighting game in which your players control giant monsters that engage in combat in the middle of a huge city. You can use both

game profiling and game object profiling tools to analyze the performance of sound in your game.

You can use game profiling tools to analyze the following:

- How the many sounds associated with monsters, police, and bystanders use the platform's streaming capabilities.
- How and when background noises such as collapsing buildings fall into virtual voice.
- Which effect plug-ins are applied to the different monster growls and how these affect CPU usage.

You can use game object profiling tools to analyze the following:

- How the attenuation radius of the sounds for each monster in your game interacts with that of each other monster.
- Where game objects such as police helicopters move relative to one another and to the monsters.
- How an RTPC such as “Panic” affects the playback of sounds associated with bystander game objects.

In this way, the game profiling and game object profiling tools can give you a complete view of your game's soundscape in action.

Exploring the Game Profiling Views

Wwise's game profiling tools consist of three views which work together to help you analyze your project's performance in detail. These views have been combined into the Profiler layout.



Tip

In Wwise, you can switch to the Profiler layout by pressing F6.

Capture Log

The Capture Log can capture and record all information coming from the sound engine. Using the Capture Log Filter, you decide what information is displayed in the Capture Log.

Capture Log

Filter... Save Log... Wwise Object Column: Wwise Object Name

Timestamp	Type	Description	Wwise Object	Game Object	Scope
02:13:30.752	Notification	End Reached And Continue	mgun_short_bu...	Transport/Sou...	Game Object
02:13:31.157	Event	Manual Event Triggered		Transport/Sou...	Game Object
02:13:31.157	Action Triggered	Manual Action	No Element	Transport/Sou...	Game Object
02:13:31.178	Notification	Play	404zone	Transport/Sou...	Game Object
02:13:31.456	Notification	Delay Started		Transport/Sou...	Game Object
02:13:31.456	Notification	End Reached And Continue	explode2	Transport/Sou...	Game Object
02:13:31.477	Notification	Delay Ended		Transport/Sou...	Game Object
02:13:31.498	Notification	Play Continue	mgun_burst2_1...	Transport/Sou...	Game Object
02:13:31.498	Notification	End Reached And Continue	New Sound	Transport/Sou...	Game Object
02:13:31.541	Notification	Play Continue	mortar_shell_in...	Transport/Sou...	Game Object
02:13:31.584	Error	Voice Starvation			
02:13:31.626	Notification	Delay Started		Transport/Sou...	Game Object
02:13:31.626	Notification	End Reached And Continue	mgun_Medium_...	Transport/Sou...	Game Object
02:13:31.840	Notification	Delay Started		Transport/Sou...	Game Object
02:13:31.840	Notification	End Reached And Continue	404zone	Transport/Sou...	Game Object
02:13:31.946	Notification	End Reached And Continue	New Sound	Transport/Sou...	Game Object
02:13:31.989	Notification	Play Continue	mortar_shell_in...	Transport/Sou...	Game Object
02:13:32.501	Notification	Delay Ended		Transport/Sou...	Game Object
02:13:32.501	Notification	Play Continue	explode1_01	Transport/Sou...	Game Object
02:13:32.565	Notification	Delay Ended		Transport/Sou...	Game Object
02:13:32.565	Notification	Play Continue	explode1_01	Transport/Sou...	Game Object
02:13:32.586	Notification	End Reached And Continue	mortar_shell_in...	Transport/Sou...	Game Object
02:13:32.629	Notification	Play Continue	mortar_explode3	Transport/Sou...	Game Object
02:13:33.034	Notification	End Reached And Continue	mortar_shell_in...	Transport/Sou...	Game Object
02:13:33.077	Notification	Play Continue	mortar_explode1	Transport/Sou...	Game Object
02:13:33.269	Notification	Delay Started		Transport/Sou...	Game Object
02:13:33.269	Notification	End Reached And Continue	mgun_burst2_1...	Transport/Sou...	Game Object
02:13:34.037	Notification	Delay Ended		Transport/Sou...	Game Object
02:13:34.037	Notification	Play Continue	mgun_burst2_1...	Transport/Sou...	Game Object
02:13:34.186	Notification	Delay Ended		Transport/Sou...	Game Object
02:13:34.186	Notification	Play Continue	mgun_Short_bu...	Transport/Sou...	Game Object
02:13:34.656	Notification	Delay Started		Transport/Sou...	Game Object
02:13:34.656	Notification	End Reached And Continue	explode1_01	Transport/Sou...	Game Object
02:13:34.698	Notification	Delay Started		Transport/Sou...	Game Object
02:13:34.698	Notification	End Reached And Continue	explode1_01	Transport/Sou...	Game Object
02:13:34.954	Notification	End Reached And Continue	mortar_explode3	Transport/Sou...	Game Object
02:13:34.976	Notification	Play Continue	New Sound	Transport/Sou...	Game Object
02:13:35.210	Notification	Delay Ended		Transport/Sou...	Game Object
02:13:35.210	Notification	Play Continue	mgun_Short_bu...	Transport/Sou...	Game Object
02:13:35.381	Notification	Delay Ended		Transport/Sou...	Game Object
02:13:35.381	Notification	Play Continue	mgun_Short_bu...	Transport/Sou...	Game Object
02:13:35.445	Notification	Delay Started		Transport/Sou...	Game Object
02:13:35.445	Notification	End Reached And Continue	mgun_Short_bu...	Transport/Sou...	Game Object
02:13:35.744	Notification	End Reached And Continue	mortar_explode1	Transport/Sou...	Game Object

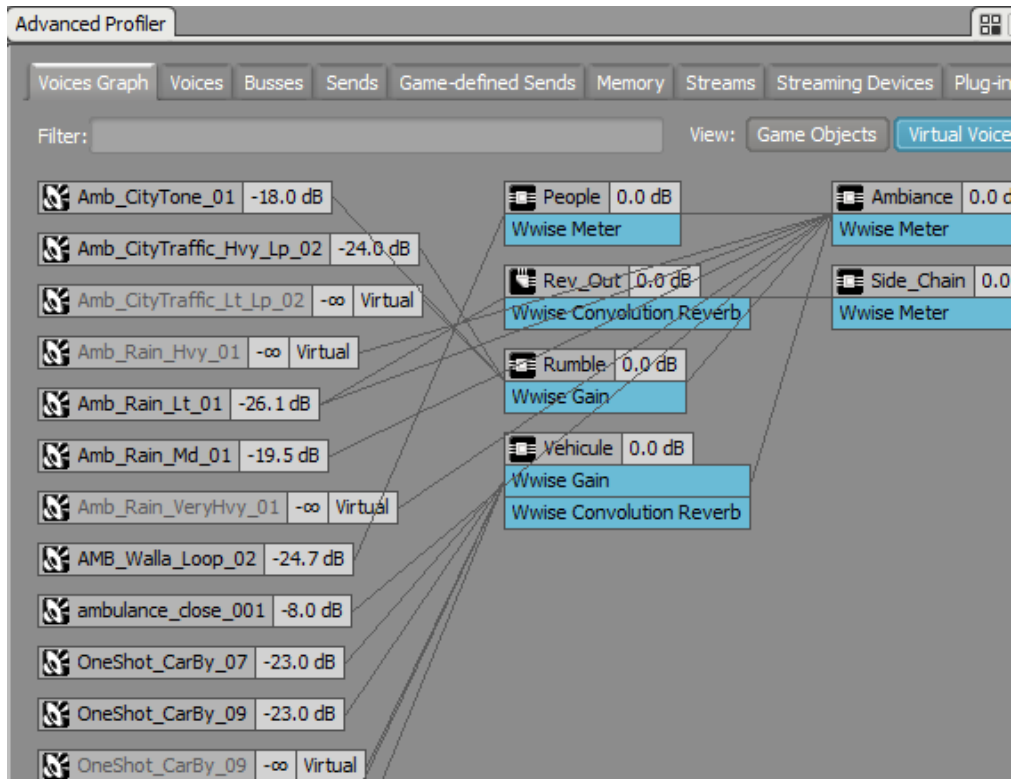
Advanced Profiler

The Advanced Profiler contains performance-related information, such as CPU, memory, and bandwidth, for each activity performed by the sound engine. The information is displayed in real time as it is captured from the sound engine. You decide what types of information are displayed in the tabs of the Advanced Profiler in the Profiler Settings dialog box.



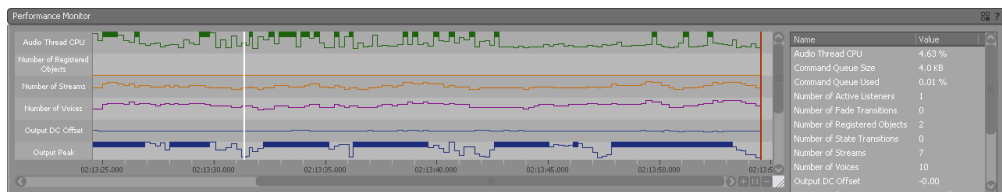
Note

Only mixing busses are displayed in the **Voices Graph** and the **Busses** tabs. Refer to [Mixing Versus Non-Mixing Busses](#) for more information on the distinction between these two types of busses.



Performance Monitor

The Performance Monitor is a comprehensive set of sound engine metrics that can help you monitor performance and troubleshoot problems. You can configure the Performance Monitor to display only the information you require.



Exploring the Game Object Profiling Views

Wwise's game object profiling tools include three views which work together to help you monitor game objects and their effect on the audio and motion in your game. You can access these views all at once in the Game Object Profiler layout.



Tip

In Wwise, you can switch to the Game Object Profiler layout by pressing F12.

Game Object Explorer

The Game Object Explorer is the control center for the Wwise game object profiling tools. Once a game object or listener is registered in game by the audio programmer, you can 'watch' them in real time in the Game Object 3D Viewer, the Game Sync Monitor, and the graph views of the RTPC tab and the Attenuation Editor. As soon as you connect to your game, the Game Objects list becomes populated with all game objects currently registered in your game.



Note

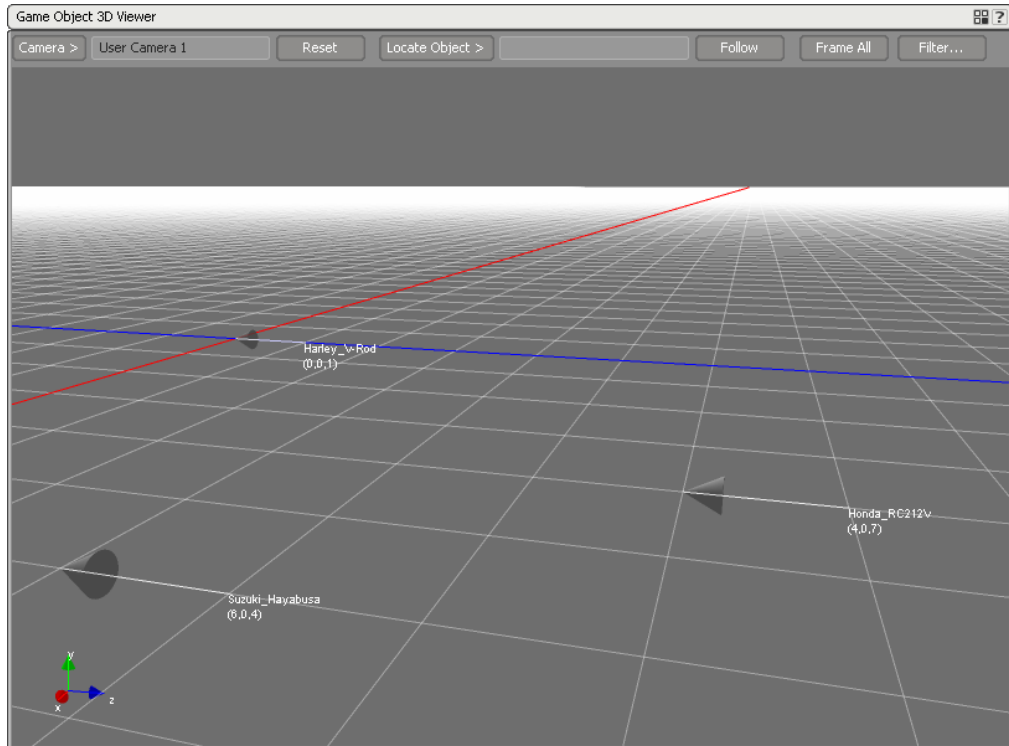
For most of these views, you need to select the objects and listeners you want to display in the Watches tab.

The screenshot shows the 'Game Object Explorer' window with two tabs: 'Game Objects' and 'Watches'. The 'Game Objects' tab is active, displaying a table with the following data:

Name	ID	Last Registered	Last Unregistered	Is Alive
Harley_V-Rod	1	00:28:33.237	00:28:25.578	✓
Suzuki_Hayabusa	2	00:28:42.581	00:28:24.064	✓
Honda_RC212V	3	00:29:47.392	00:22:40.298	✓
Transport/Soundcaster	0	-	-	✓

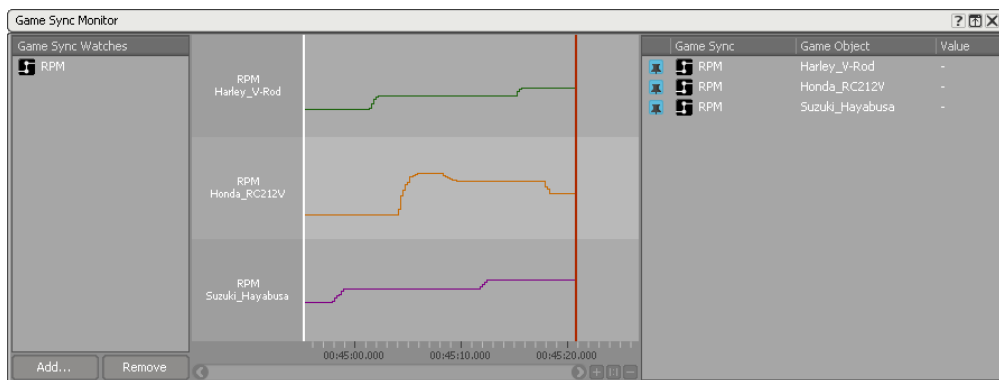
Game Object 3D Viewer

The Game Object 3D Viewer is a three-dimensional visual representation of game objects and listeners. You can configure the Game Object 3D Viewer to display different types of information, as well as to change how game objects and listeners themselves are displayed.



Game Sync Monitor

The Game Sync Monitor is a tool for analyzing RTPC values in real time. During gameplay, graphs are drawn for the RTPC values that change for watched game objects.



Connecting to a Local/Remote PC or Game Console

If you want to simulate different sounds or motion FX in game, or want to profile and troubleshoot different aspects of your game on a particular platform, you need to first connect to the PC or game console upon which the game runs. You can connect to any Wwise sound engine that is running and available on your local area network.



Caution

When profiling, it is recommended that you connect to the Profile build configuration of the Wwise sound engine, even from the Debug build of your game, because the performance of the Debug configuration has not been optimized.

To help you find the PC or game console you are looking for, Wwise automatically searches for all PCs and game consoles on the same subnet of the network that are currently running a version of the Wwise sound engine. You can also connect to consoles or PCs outside your subnet by manually entering the IP address of the platform.



Note

If your game doesn't appear in the Remote Connections dialog box in Wwise, refer to [Troubleshooting Remote Connection Issues](#) to troubleshoot the problem.

To connect to a local/remote PC or game console:

1. On the toolbar, click the **Remote** button.

The Remote Connection dialog box opens with a list of PCs and consoles on the network that are currently running a version of the Wwise sound engine.

2. From the list, select the platform to which you want to connect.



Note

The status of the console must be “Ready” before you can connect to it.

3. If you want to start capturing information from the moment you connect, select the **Start capture on connect** option.
4. Click **Connect**.

Wwise connects to the local or remote platform. The name of the PC or game console appears in the toolbar.



Note

You can manually connect to a platform by clicking **Connect To IP** and then typing the IP address of the remote console.

To disconnect from a local/remote PC or game console:

1. On the toolbar, click the **Disconnect** button.

Wwise disconnects from the PC or console.

Related Topics

- [Using the Remote Connection History List](#)
- [Capturing Data from the Sound Engine](#)
- [Specifying Network Ports](#)
- [Troubleshooting Remote Connection Issues](#)

Using the Remote Connection History List

Wwise maintains a history of all the consoles or PCs that you've successfully connected to in the past. This way you can easily find a PC or console that you've been working on. Before connecting to a PC or console, you must first verify its status. The status of a PC or console can be one of the following types:

- **Ready** - When it is ready to accept a connection.
- **Busy** - When it is already connected to a machine and is therefore not accepting any connections.
- **Different Version** - When the version of Wwise is not compatible with the version of the game running on the remote PC or console.
- **Not Available** - When it is no longer connected to the network.

You can clean up the history when you no longer want PCs or consoles to appear in your history list.

To connect to a PC or console using the history list:

1. On the toolbar, click the **Remote** button.

The Remote Connection dialog box opens with a list of PCs and consoles that are currently running on the network.

2. Switch to the History tab.
3. From the History list, select the PC or console to which you want to connect.

4. If you want to start capturing information from the moment you connect, select the **Start capture on connect** option.
5. Click **Connect**.

Wwise connects to the local/remote PC or game console. The name of the PC or game console appears in the toolbar.

To clean up your history list:

1. On the toolbar, click the **Remote** button.

The Remote Connection dialog box opens with a list of PCs and consoles that are currently running on the network.

2. Switch to the History tab.
3. From the History list, select the PC or console that you want to remove.
4. Do one of the following:

Click **Remove**.

Press the **Delete** key.

The PC or console is removed from the History list.

Related Topics

- [Connecting to a Local/Remote PC or Game Console](#)
- [Capturing Data from the Sound Engine](#)
- [Troubleshooting Remote Connection Issues](#)

Capturing Data from the Sound Engine







After connecting to a PC or game console, you can begin to profile the audio and motion FX in your game by capturing data directly from the sound engine. All the information coming from the sound engine is displayed in the Capture Log. An entry is recorded in the Capture Log for the following types of information:

Notifications	Properties	Banks
Markers	States	Errors
Events	Switches	Messages
Actions	Prepared Events	

You can monitor each of these entries using the Performance Monitor and Advanced Profiler. These views contain detailed information about memory,

voice, and effect usage, streaming, SoundBanks, plug-ins, and so on. For more information on monitoring performance, refer to [Monitoring and Troubleshooting with the Performance Monitor](#). You can also get Wwise to parse the data captured and provide you with some statistics on certain audio elements in your game or project. For more information on gathering statistics, refer to [Gathering Statistical Information from a Capture Session](#).

Wwise uses the following special indicators and color to help you quickly sort through the many entries that can appear in the Capture Log.

Interface Element	Description
Indicators	
White/gray circles 	The white/gray circles indicate which items in the Capture Log were captured within 100 ms of the Performance Monitor time cursor position. The whiter the indicator, the closer the entry is to the time cursor's position. You can force the cursor to the timestamp of a specific log entry by shift-clicking on the entry.
Blue circles 	The blue circles indicate which items in the Capture Log are related to each other. The blue circles are displayed when you select an entry in the log.
Colors	
Selected Entry 	The entry that is selected in the Capture Log is highlighted in blue.
Related Entry 	All entries that are related to the selected entry are highlighted in teal blue.
Errors 	<p>All errors that occur during the capture process are highlighted in yellow.</p> <p>The Voice Starvation error message occurs when the sound engine cannot provide audio or motion data to the platform hardware buffer in a timely manner. This type of problem occurs when there is excessive use of the host CPU, where the audio thread CPU is near 100%. For example, it can occur when the platform CPU is trying to mix too many sources that are using too many audio effects simultaneously.</p> <p>The Source Starvation error message occurs when the input stream starves most likely because the throughput obtained from the stream manager is insufficient for some reason. For example, it can occur when there are long seek times on the DVD.</p>
Messages 	All messages are highlighted in green.

The following illustration shows how the different indicators and colors are used in the Capture Log.

Any errors are highlighted in yellow.

White/gray indicators show entries captured within 100ms of the Performance Monitor time cursor.

Blue indicators show which entries are related to one another. Related entries are also highlighted in teal blue.

All messages are highlighted in green.

Timestamp	Type	Description	Wwise Object	Game Object
00:00:07.081	Bank	Bank Load Request Received (from Init.bnk)	Init	
00:00:07.081	Bank	Bank Loaded (from Init.bnk)	Init	
00:00:07.081	Bank	Bank Load Request Received (from main.bnk)	main	
00:00:07.097	Bank	Bank Loaded (from main.bnk)	main	
00:00:07.113	Message	Main Character walks.		
00:00:07.113	Message	Unknown Event name : S_FOOTOO		
00:00:07.113	Error	ID not found		
00:00:07.144	Event	Event Triggered	S_FOOTOO	External Auth...
00:00:07.144	Action Triggered	Play	footsteps	External Auth...
00:00:07.160	Notification	Play	F5_Concrete_04	External Auth...
00:00:07.237	Event	Event Triggered	S_FOOTOO	External Auth...
00:00:07.237	Action Triggered	Play	footsteps	External Auth...
00:00:07.253	Notification	Play	F5_Concrete_02	External Auth...
00:00:07.347	Event	Event Triggered	S_FOOTOO	External Auth...
00:00:07.347	Action Triggered	Play	footsteps	External Auth...
00:00:07.375	Notification	Play	F5_Concrete_01	External Auth...
00:00:07.440	Event	Event Triggered	S_FOOTOO	External Auth...
00:00:07.440	Action Triggered	Play	footsteps	External Auth...
00:00:07.452	Notification	Play	F5_Concrete_03	External Auth...
00:00:07.542	Event	Event Triggered	S_FOOTOO	External Auth...
00:00:07.542	Action Triggered	Play	footsteps	External Auth...
00:00:07.558	Notification	Play	F5_Concrete_04	External Auth...
00:00:07.643	Event	Event Triggered	S_FOOTOO	External Auth...
00:00:07.643	Action Triggered	Play	footsteps	External Auth...
00:00:07.659	Notification	Play	F5_Concrete_02	External Auth...
00:00:07.729	Notification	Event Finished	F5_Concrete_04	External Auth...
00:00:07.737	Message	rtpdstpos=FRONT, Sound plays in the BACK		
00:00:07.747	Event	Event Triggered	S_FOOTOO	External Auth...
00:00:07.747	Action Triggered	Play	footsteps	External Auth...
00:00:07.764	Notification	Play	F5_Concrete_01	External Auth...
00:00:07.851	Event	Event Triggered	S_FOOTOO	External Auth...
00:00:07.851	Action Triggered	Play	footsteps	External Auth...
00:00:07.879	Notification	Play	F5_Concrete_03	External Auth...
00:00:07.934	Notification	End Reached	F5_Concrete_01	External Auth...
00:00:07.934	Notification	Event Finished		
00:00:07.952	Event	Event Triggered	S_FOOTOO	External Auth...
00:00:07.952	Action Triggered	Play	footsteps	External Auth...
00:00:07.952	Notification	End Reached	F5_Concrete_02	External Auth...
00:00:07.952	Notification	Event Finished		
00:00:07.973	Notification	Play	F5_Concrete_01	External Auth...
00:00:08.061	Event	Event Triggered	S_FOOTOO	External Auth...

You can also profile some of your ideas or prototypes even before they have been integrated in the game by capturing directly from the Soundcaster in Wwise. For more information on using the Soundcaster, refer to [Chapter 31, Creating Simulations](#).



Note

Before profiling your game, you should load all unloaded work units back into the project. When work units are unloaded, the profiling information may be incomplete and possibly misleading.

Specifying the Type of Information That Will be Captured

Because of the potentially large amount of data being transferred, capturing from the sound engine may affect Wwise's performance. Therefore, you may want to limit the type of information that is being generated by the Advanced Profiler by using the Profiler Settings dialog box. By deselecting one or more of the information types, you can:

- Save network transfer bandwidth.
- Save memory space in Wwise.

- Save CPU time in game by not calculating the data.
- Save CPU time in Wwise by not processing or drawing the data.

You can also specify whether you want Wwise to dump the entire contents of a capture session, from all views of the Profiler and Game Object Profiler layouts, to a profiling session file. The data within this file can be loaded back into Wwise at a later time for further analysis of the audio in your game.



Note

Wwise overwrites this file each time a new capture session begins, so if you want to save older capture sessions, you will have to rename the file after each profiling session.

To specify the type of information that will be generated and captured:

1. Do one of the following:

In the Advanced Profiler title bar, click the View Settings icon.

From the menu bar, click **Project > Profiler Settings**.

Press **Alt+G**.

The Profiler Settings dialog box opens.

2. Select one or more of the following information types:

Plug-in Data to capture information related to the various plug-ins.



Note

The current version of Wwise does not support the capture of CPU usage data for plug-ins on the PlayStation®3. Instead of a value, a dash (-) is displayed under % CPU.

Memory Data to capture information related to the memory pools registered in the sound engine's Memory Manager.

Stream Data to capture information related to the streams managed by the sound engine.

Voices Data to capture information related to each of the voices managed by the sound engine.

Auxiliary Sends Data to capture information related to auxiliary sends.

Listener Data to capture information related to each of the listeners managed by the sound engine.

Obstruction/Occlusion Data to capture information related to the obstruction and occlusion affecting game objects.

Markers Notification Data to capture information related to marker notifications in audio files and custom cue notifications in music segments.

Output Data to capture information related to output peak and output DC offset.

Wii Remote™ Data to capture information related to speakers in Wii remote controls.

SoundBanks to capture information related to the SoundBanks that have been loaded into memory.

Loaded Media to capture information related to the media loaded into memory.

Prepared Events to capture information related to events that have been prepared using the PrepareEvent function.

Prepared Game Syncs to capture information related to game syncs that have been prepared using the PrepareGameSyncs function.

Motion Data to capture motion information that is managed by the sound engine.

Music Transitions to capture information related to the transitions that occur between music objects.



Note

When you deselect one of the information types, the corresponding tab is removed from the Advanced Profiler.

3. Under **Capture Log**, type the maximum amount of memory to be used. The minimum size allowed is 10 MB, and the maximum is 1000 MB.

The Game Profiler will use no more than the amount of memory you specify for the capture log.

4. **Number of sessions kept** lets you specify the number of profiling sessions kept on disk. When more files exist, the oldest is deleted.
5. Click OK.

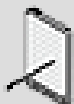
The Game Profiler will only capture data for the information types that you selected.

Related Topics

- [Starting/Stopping the Capture Process](#)
- [Filtering the Capture Log](#)
- [Sorting the Capture Log](#)
- [Gathering Statistical Information from a Capture Session](#)
- [Saving the Capture Log](#)
- [Loading the Data from a previous Remote Capture Session](#)

Starting/Stopping the Capture Process

When you start capturing information from the sound engine, data is automatically recorded in the Capture Log, Performance Monitor, and Advanced Profiler. You can review the information as it is being captured, or you can stop the capture to investigate issues as they arise. If you plan to review the information as it is being captured, you can use the Follow Capture Time option to keep the Capture Log and Performance Monitor in sync with the capture time.



Note

Each time you start a capture process, Wwise clears the Capture Log. If you want to keep the information that is in the Capture Log, you can save it to a .prof file or a text file. For more information on saving the contents of the Capture Log, refer to [Saving the Capture Log](#).

To start the capture process:

1. Do one of the following:

From the menu bar, click **Layouts > Profiler**.

Press **F6**

The Profiler layout is displayed.

2. If you want to monitor the performance as the data is being captured, click the **Follow Capture Time** button on the toolbar.

When this option is selected, the following occurs:

The Capture Log will scroll automatically as the data is being captured.

The Performance Monitor time cursor will follow the game time cursor.

3. On the toolbar, click the **Start Capture** button to begin capturing data in the Capture Log.
4. To stop capturing, click **Stop Capture**.

Related Topics

- [Connecting to a Local/Remote PC or Game Console](#)
- [Filtering the Capture Log](#)
- [Sorting the Capture Log](#)
- [Gathering Statistical Information from a Capture Session](#)
- [Saving the Capture Log](#)
- [Loading the Data from a previous Remote Capture Session](#)
- [Monitoring and Troubleshooting with the Performance Monitor](#)

Filtering the Capture Log

You may find that there is too much information being displayed in the Capture Log. If this is the case, you can filter the Capture Log to display the specific information you are looking for. For example, you can display only the information related to a particular game object, or only event-related information or state-related information.

To filter the Capture Log:

1. In the Capture Log, click **Filter**.

The Capture Log Filter dialog box opens.

2. Filter the capture log by selecting only those options that you want to appear in the Capture Log.



Note

For a description of each of the options click the Help icon in the Capture Log Filter dialog box.

3. Click OK.

The Capture Log is filtered according to the criteria you selected.

Related Topics

- [Starting/Stopping the Capture Process](#)
- [Sorting the Capture Log](#)
- [Gathering Statistical Information from a Capture Session](#)
- [Saving the Capture Log](#)
- [Loading the Data from a previous Remote Capture Session](#)

Sorting the Capture Log

You can sort the information in the Capture Log by any of the column headings. For example, you may want to sort the entries by type, Wwise object, or game object.



Note

You can't sort the information in a column while Wwise is capturing data from the sound engine.

To sort the Capture Log:

1. In the Capture Log, click a column header.

The entries are sorted by the column header you selected.

Related Topics

- [Starting/Stopping the Capture Process](#)
- [Filtering the Capture Log](#)
- [Gathering Statistical Information from a Capture Session](#)
- [Saving the Capture Log](#)
- [Loading the Data from a previous Remote Capture Session](#)

Gathering Statistical Information from a Capture Session

At any point during or after a capture session, you can have Wwise parse the information captured to provide you with statistics about certain audio elements in your Wwise project or game.



Note

Currently only information about the dynamic dialogue elements can be extracted from the capture session. To completely understand the statistical information presented, it is important that you understand the different steps involved in generating dynamic dialogue with Wwise. For more information on dynamic dialogue, refer to [Understanding the Dynamic Dialogue System](#).

The Profiler Statistics view displays the statistical information extracted from the capture session. It contains a series of pre-defined queries that provide you with statistics on different subsets of information within the capture log. For example, you can determine which paths were resolved during the capture session, how many times they were resolved, and how many times this resulted in audio being played back. You can also filter the information to focus your search, find any issues, and determine what actions are required, if any.

You can extract statistics from any 'active' capture session, which means you can gather statistics on data captured remotely from a game or locally from within Wwise. A capture session remains active as long as the information is displayed in the Capture log.

To gather statistical information from a capture session:

1. Start a local or remote capture session.
2. From the menu bar, click **Views > Profiler Statistics**.

The Profiler Statistics view opens.

3. From the Queries list, select one of the following queries:

Paths Used - Shows you which paths were resolved and added to the dynamic sequence list during the capture session, how many times they were resolved, and how many times this resulted in the playback of audio.

Paths Added - Shows you which dialogue events were triggered during the capture session. For each dialogue event triggered, it displays the corresponding audio object that was added to the dynamic sequence list, as well as how many times it was added.

Dialogue Events Distribution - Displays statistical information about the dialogue events that were triggered during the capture session. Specifically, it shows you how many times each dialogue event was resolved versus how many times it actually resulted in the playback of an audio object.

4. Click **Run**.

The results of the query are displayed in the Results pane.

5. Use one or more filter options to filter out the information you don't need. This allows you to focus your search so you can quickly determine if any actions are necessary.

Related Topics

- [Starting/Stopping the Capture Process](#)
- [Filtering the Capture Log](#)
- [Sorting the Capture Log](#)
- [Saving the Capture Log](#)
- [Loading the Data from a previous Remote Capture Session](#)

Saving the Capture Log

Wwise clears the Capture Log each time you start a capture process, so if you want to keep the information in the Capture Log, you will have to save it to a file. You can save the file as a .prof file (default) or a text file. A .prof file can be loaded into the Wwise profiler by using the Remote button on top of the window and then Connect As IP. The information will be displayed as if you had just profiled your game. If a text file is preferable, Wwise saves only the information from the Capture Log. The current columns and sorting will be intact. If you want to sort or filter the information further, you can always import the text file into an external spreadsheet program.

To save the capture log:

1. In the Capture Log, click **Save Log**.

The Save As window opens.

2. Navigate to the location where you want to save the file.
3. In the File name text box, type the name of the file. Choose the .prof or .txt extension.
4. Click **Save**.

The contents of the Capture Log are saved as a text file.

Related Topics

- [Starting/Stopping the Capture Process](#)
- [Filtering the Capture Log](#)
- [Sorting the Capture Log](#)
- [Gathering Statistical Information from a Capture Session](#)

- [Loading the Data from a previous Remote Capture Session](#)
- [Monitoring and Troubleshooting with the Performance Monitor](#)

Loading the Data from a previous Remote Capture Session

While you are capturing information from your game, Wwise dumps the information into a profiling session file called ProfilingSession.prof. You can load this information back into Wwise at a later time to perform further analysis of the audio in your game.

Each new profiling session will create a new file, with an incrementing number after the file name. You can specify the number of files kept in the Profiler Settings.



Note

If you want Wwise to save the data from a capture session to a file, you can click the Save Log button on top of the Capture Log. This will copy the latest session to your new location.

To load the data from a previous capture session:

1. Switch to the Profiler layout by doing one of the following:

From the menu bar, click **Layouts > Profiler**.

Press **F6**.

2. From the toolbar, click the **Remote** button.

The Remote Connections dialog box opens.

3. Click the **Connect To File** button.
4. Select one .prof file and click **Open**.

All the data collected in the capture session is loaded into the views of the Profiler and Game Object Profiler layouts.

To compute statistics from multiple profiling sessions

1. Open the Profiler Statistics View from the **View** menu.
2. From the toolbar, click the **Remote** button.

The Remote Connections dialog box opens.

3. Click the **Connect To File** button.
4. Select as many .prof files as you need and click **Open**.

All the data collected in the selected capture sessions is processed and the resulting statistics are presented in the **Profiling Statistics** view.

Related Topics

- [Starting/Stopping the Capture Process](#)
- [Filtering the Capture Log](#)
- [Sorting the Capture Log](#)
- [Gathering Statistical Information from a Capture Session](#)
- [Saving the Capture Log](#)
- [Monitoring and Troubleshooting with the Performance Monitor](#)

Monitoring and Troubleshooting with the Performance Monitor

Since Wwise is so tightly integrated with the sound engine, you can monitor a wide range of key performance indicators in real time as the game is being played. The Game Profiler also works with the Soundcaster, Game Simulator, and other SoundFrame applications so that you can monitor the performance of your ideas or prototypes even before they have been integrated into the game.

As Wwise captures information related to the activities of the sound engine, performance graphs are created in real time in the Performance Monitor. The actual numbers and percentages related to the graphs are also displayed in the Performance Data list, which is located to the right of the graph view.

The different graphs will help you quickly locate areas in your game where the audio or motion is surpassing certain constraints, such as platform limitations. Using a combination of the Performance Monitor, Capture Log, and Advanced Profiler, you can troubleshoot any issues that may arise.

The Performance Monitor is customizable, which means that you can decide which performance indicators or counters are displayed in the Performance Monitor. For more information on customizing the Performance Monitor, refer to [Customizing the Performance Monitor](#).



Tip

You can monitor the performance as the data is being captured, or you can perform a capture session and then scroll through the graph to investigate the issues afterwards. If you plan to monitor the performance while the data is being captured, you must select

the Follow Capture Time option to keep the Capture Log and Performance Monitor in sync with the capture time.

To monitor performance:

1. Connect to a remote console, PC, SoundFrame application, or running version of the Game Simulator.



Note

If you are profiling a certain model or prototype using the Soundcaster, the information is captured as you play back the different modules.

2. Do one of the following:

From the menu bar, click **Layouts > Profiler**.

Press **F6**

The Profiler layout is displayed.

3. On the toolbar, click the **Start Capture** button.

Wwise begins to graph the performance of the particular platform in relation to the information captured from the sound engine.



Note

Solid blocks indicate places where the captured information exceeds a minimum or maximum value. Blocks at the top of a graph indicate values higher than the maximum, and blocks at the bottom indicate values lower than the minimum.

4. Drag the Performance Monitor time cursor (white) to a location on the graph where there is a performance issue.



Note

As you scroll through the graph view, the position of the Capture Log and the information in the Performance Data pane and the Advanced Profiler are automatically updated.

5. Use the time cursor indicators (white circles) in the Capture Log to locate the event, action, or other activities performed by the sound engine that are

creating the issue. Alternatively, shift-click on entry in the Capture Log to synchronize all the profiling views on this timestamp.

6. To edit the contents and/or properties of an object or event, double-click the entry in the Capture Log. The corresponding event or object will open in the Event Editor or Property Editor where you can make any modifications that are necessary.
7. Review the information on the various tabs of the Advanced Profiler to get more information about the performance of the particular activity.
8. To save a text file version of the performance monitor graph, right-click the graph and select **Save All Counters to File**. This will allow you to save the counters to a tab-delimited text file that includes all counters from the beginning of the capture to the current moment, recorded at 200 ms intervals.

Related Topics

- [Customizing the Performance Monitor](#)
- [Starting/Stopping the Capture Process](#)
- [Saving the Capture Log](#)

Customizing the Performance Monitor

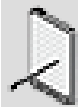
In some cases, you may want to monitor several key indicators at the same time and in other cases, you may want to focus on one particular area. To meet the needs of each monitoring session, you can add and remove counters from the Performance Monitor.

To modify the Performance Monitor display:

1. In the Performance Monitor title bar, click the View Settings icon.

The Performance Monitor Settings dialog box opens.

2. In the Show In Graph column, select the counters that you want to display in the Performance Monitor graph view.



Note

For a description of each counter, click the Help icon in the Performance Monitor Settings dialog box.

3. In the Show In List column, select the counters that you want to display in the Performance Data list.
4. Set a minimum and maximum value for the counters you selected. The min and max values define the vertical range of the graph band for each counter.



Note

The current version of Wwise does not support the capture of CPU usage data for plug-ins on the PlayStation®3. A dash (-) is displayed beside **Total Plug-in CPU** instead.

5. Click OK.

The Performance Monitor is updated to display only the counters that you selected.



Note

If you select a type of data that has been disabled in the Profiler Settings, the Performance Monitor will not display a graph for that heading and will display the heading itself in red. For more information on changing Profiler settings, refer to [Specifying the Type of Information That Will be Captured](#).

Keeping Track of Objects and Listeners with the Game Object Explorer

The Game Object Explorer is the starting point for studying game objects and listeners. Within this view, you can see all the registered game objects in your game, as well as control which game objects and listeners will be watched by Wwise. The game objects that you have selected for 'watching' become visible in the Game Sync Monitor, and both game objects and listeners show up in the Game Object 3D Viewer and the RTPC graph and the Attenuation Editor.

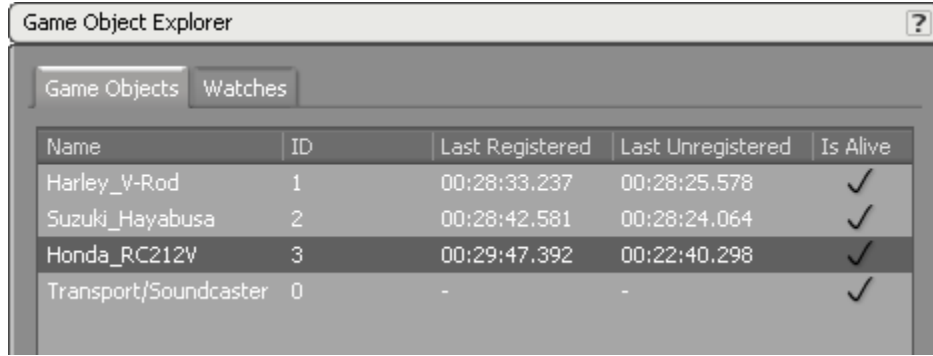
Keeping track of game objects and listeners with the Game Object Explorer involves the following:

- [Understanding the List of Game Objects](#)
- [Monitoring Game Objects and Listeners with Watches](#)

Understanding the List of Game Objects

The Game Objects tab of the Game Object Explorer contains a list of all game objects that are or have been active during this session. Game objects are created and destroyed by the audio programmer and it is the programmer who supplies the name and ID for each game object as they are created.

After connecting to your game, this list will update itself automatically.



The screenshot shows a window titled "Game Object Explorer" with a help icon in the top right. Below the title bar are two tabs: "Game Objects" and "Watches". The "Watches" tab is selected, displaying a table with the following data:

Name	ID	Last Registered	Last Unregistered	Is Alive
Harley_V-Rod	1	00:28:33.237	00:28:25.578	✓
Suzuki_Hayabusa	2	00:28:42.581	00:28:24.064	✓
Honda_RC212V	3	00:29:47.392	00:22:40.298	✓
Transport/Soundcaster	0	-	-	✓

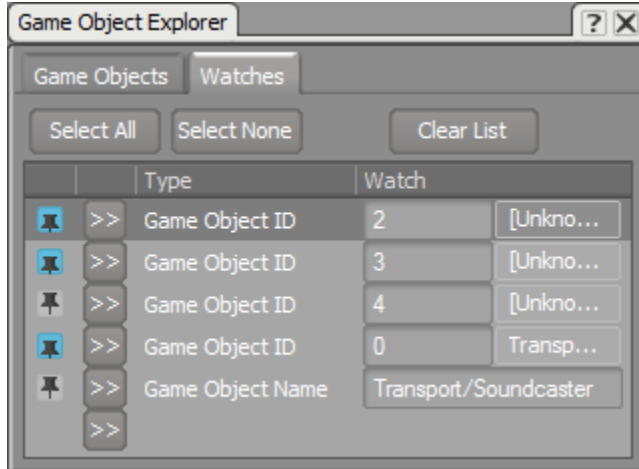
Game objects that are currently active are labeled as “Alive”. Game objects become alive when they are registered, and cease being alive when they are unregistered. For example, in a first-person shooter, a game object may be considered no longer alive if it is destroyed or moves beyond the range the player can detect in the game geometry. On the other hand, in a real-time strategy game, all game objects on the map may remain alive until they are destroyed.

Related Topics

- [Monitoring Game Objects and Listeners with Watches](#)
- [Defining Game Object Watches](#)
- [Editing and Deleting Watches](#)

Monitoring Game Objects and Listeners with Watches

Watches are individual processes you can use to monitor the game objects and listeners in your game. When you assign a watch to a game object or listener, the watch reports back on a regular basis with details about that item's status. The information retrieved through these watches can be used by the Game Object 3D Viewer, the RTPC tab graph, the Attenuation Editor, and the Game Sync Monitor. If you don't watch any game objects, you won't see any data in the game object profiling views.



Game object watches can be based on either name or ID. Multiple game objects can share the same name within a game, but object IDs are unique. Therefore, if you want to create one watch that monitors many identical objects (such as drones), use a name-based watch. On the other hand, if you want to make sure to monitor only one particular object, create an ID-based watch.

Working with watches can involve the following tasks:

- [Defining Game Object Watches](#)
- [Defining Listener Watches](#)
- [Editing and Deleting Watches](#)
- [Selecting Watches for Display](#)

Defining Game Object Watches

There are two ways to define a game object watch:

- From the game object list.
- From the watch list.

To define a game object watch from the game object list:

1. In the Game Objects tab of the Game Object Explorer, right-click a game object and select **Add Game Object to Game Object Watch List (By ID or By Name)**.

A watch based on that game object's unique ID or name is added to the watch list.

To define a game object watch from the watch list:

1. In the Game Object Explorer, switch to the Watches tab.
2. Click the Selector button (>>) and select one of the following options:

Game Object Name to create a game object watch based on an object's name.

Game Object ID to create a game object watch based on an object's ID.

Global Game Object to create a watch on the global game object. (Mostly useful when watching values that will be used globally or on busses).

3. In the Watch column, enter the name or ID on which you want the watch to be based and press the **Enter** key.

The watch you have defined is added to the watch list.



Note

To remove a watch from the watch list, select the watch and press the Delete key. To remove the entire watch list you created to begin again, click **Clear List**.

Related Topics

- [Monitoring Game Objects and Listeners with Watches](#)
- [Defining Listener Watches](#)
- [Editing and Deleting Watches](#)
- [Selecting Watches for Display](#)

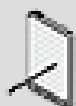
Defining Listener Watches

Defining a listener watch is similar to defining a game object watch.

To define a listener watch:

1. In the Game Object Explorer, switch to the Watches tab.
2. Click the Selector button (>>) and select **Listener ID** to create a listener watch based on a listener's ID.
3. In the Watch column, enter the ID on which you want the watch to be based and press the **Enter** key.

The listener watch you have defined is added to the watch list.



Note

To remove a watch from the watch list, select the watch and press the Delete key. To remove the entire watch list you created to begin again, click **Clear List**.

Related Topics

- [Monitoring Game Objects and Listeners with Watches](#)
- [Defining Game Object Watches](#)
- [Editing and Deleting Watches](#)
- [Selecting Watches for Display](#)

Editing and Deleting Watches

You can easily edit or delete a watch you have created. Note that when you do this, you do not affect the original game object or listener, only the watch collecting data on it.

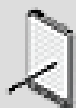
To edit a watch:

1. In the Game Object Explorer, switch to the **Watches** tab.
2. To change the type of watch, click the Selector button (>>) and select a different type of watch
3. To change the description of the watch, click in the Watches column and enter a different name or ID.

To delete a watch:

1. In the Game Object Explorer, switch to the **Watches** tab.
2. Select the watch and press the **Delete** key.

The watch is removed from the watch list.



Note

To remove the entire watch list you have created to begin again, click **Clear List**.

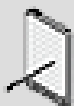
Related Topics

- [Monitoring Game Objects and Listeners with Watches](#)
- [Defining Game Object Watches](#)
- [Defining Listener Watches](#)
- [Selecting Watches for Display](#)

Selecting Watches for Display

By default, all watches are pinned, which means that the game object or listener associated with those watches will automatically be displayed in the Game

Object 3D Viewer, and the RTPC and Attenuation Editor graphs. You can, however, unpin some watches to make these views less cluttered. Displaying fewer objects also uses less CPU, which can improve the performance of the Game Object 3D Viewer.



Note

This does not affect which Game Objects are displayed in the Game Sync Monitor. All Game Objects in the Watch List are displayed in the Game Sync Monitor.

To pin or unpin a watch in the watch list:

1. In the Watches tab of the Game Object Explorer, click the pin icon to the left of the watch name.

To pin all the watches in the watch list:

1. In the Watches tab of the Game Object Explorer, click **Select All**.

To unpin all the watches in the watch list:

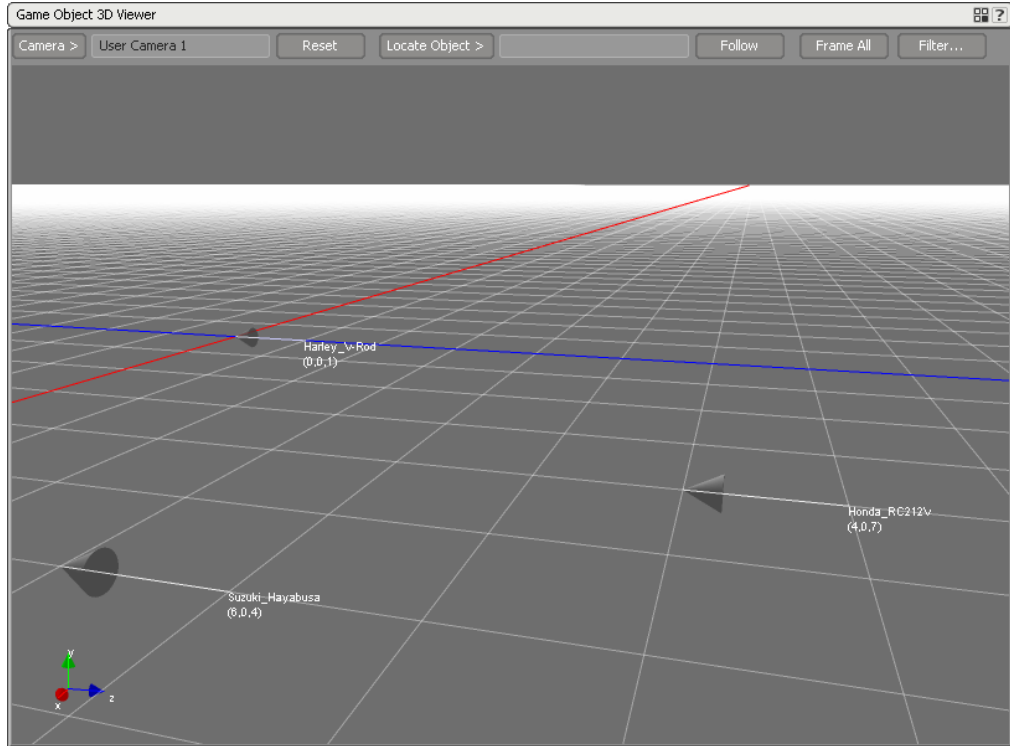
1. In the Watches tab of the Game Object Explorer, click **Select None**.

Related Topics

- [Monitoring Game Objects and Listeners with Watches](#)
- [Defining Listener Watches](#)
- [Editing and Deleting Watches](#)

Examining Objects with the Game Object 3D Viewer

The Game Object 3D Viewer takes the data from the game object and listener watches you have defined and displays them in graphical form. You can use this viewer to see how objects and listeners interact and move relative to one another in real time.



Each watched game object and listener is displayed in the Game Object 3D viewer as an arrow. Depending on the options you select in the Viewer Settings, you can also observe any cone or radius that has been applied to a game object.



Tip

You can access many of the controls in the Game Object 3D Viewer using keyboard shortcuts. For a complete list of shortcuts, refer to [Appendix B, Shortcuts](#).

Specifying the Data to be Displayed in the Game Object 3D Viewer

By changing the settings of the Game Object 3D viewer, you can have as much or as little data displayed as you choose. Naturally, the more data types you select, the more complex the view will be.

To specify the data to be displayed in the Game Object 3D Viewer:

1. In the Game Object 3D Viewer, click the Settings icon.

The Game Object 3D Viewer Settings dialog box opens.

2. Select the options that you want to appear in the Game Object 3D Viewer.



Note

For a description of each of the options click the Help icon in the Game Object 3D Viewer Settings dialog box.

3. Click **Close**.

The data is displayed in the Game Object 3D Viewer according to the criteria you selected.

Related Topics

- [Working with the Camera](#)
- [Filtering Watches](#)

Working with the Camera

You can explore the 3D viewer in all three dimensions by controlling and moving the camera with the mouse. In this way, you can observe your watched game objects and listeners as they interact with one another.

To select a game object or listener on which to center the camera:

1. In the Game Object 3D Viewer, click **Locate Object** and select the object you want the camera to center on from the list.

The object or listener you select is shown in the center of the camera's field of vision.

To assign the camera to a game object:

1. From the Locate Object list in the Game Object 3D Viewer, select a game object.
2. Click **Follow**.

The camera follows the object as it moves during capture.

To fit all game objects and listeners within the Game Object 3D Viewer:

1. In the Game Object 3D Viewer, click **Frame All**.

The viewer adjusts to include all selected game objects and listeners.

To see the game from the point of view of a listener:

1. From the Camera list in the Game Object 3D Viewer, select a listener to serve as the camera origin.

The listener you select becomes the origin point of the camera's field of vision and the alignment of the camera matches that of the listener.



Note

If you select the Follow option or use the WASD or arrow keys to change the camera position, the alignment and origin of the camera will no longer match that of the listener.

To pan the camera:

1. Ctrl+click and drag within the Game Object 3D Viewer.

The camera pans left and right, and up and down.



Note

First Person and Listener cameras do not support panning.

To zoom the camera:

1. Right-click and drag within the Game Object 3D Viewer.

The camera zooms in and out.

To rotate the camera:

1. From the Camera list in the Game Object 3D Viewer, select either User Camera 1 or User Camera 2.
2. Click and drag within the Game Object 3D Viewer.

The camera rotates around the center point.



Tip

You can switch back and forth between cameras while still maintaining your settings for each.

To operate the camera like a first-person shooter:

1. From the Camera list in the Game Object 3D Viewer, select the First Person or any one of the Listener cameras.

You can now do the following:

Press WASD or the arrow keys to move the camera.

Click and drag to orient the camera.

Hold Shift to move faster.

Right-click and drag or use the mouse center wheel to zoom in and out.

To reset the camera to its default position:

1. In the Game Object 3D Viewer, click **Reset**.

The camera returns to its default position.

Related Topics

- [Specifying the Data to be Displayed in the Game Object 3D Viewer](#)
- [Filtering Watches](#)

Filtering Watches

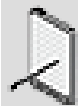
By default, the Game Object 3D Viewer shows all the game objects and listeners you are watching. If you want to show only certain watched game objects and listeners, you can filter what is displayed.

To filter the display in the Game Object 3D Viewer:

1. In the Game Object 3D Viewer, click **Filter**.

The Game Object 3D Viewer Filter dialog opens.

2. To display or hide a watch, select or deselect the pin icon.



Note

Watched game objects and listeners are only displayed in the Game Object 3D Viewer if they are pinned.

3. To display all the watches, click **Pin All**.
4. To hide all the watches, click **Unpin All**.

5. Click **Close** to return to the Game Object 3D Viewer.

Related Topics

- [Specifying the Data to be Displayed in the Game Object 3D Viewer](#)
- [Working with the Camera](#)

Evaluating Game Syncs with the Game Sync Monitor

The Game Sync Monitor is a profiling tool that allows you to track the values of RTPCs as they are applied to the game objects you are watching. These values are graphed in real time as they are received by the watches.

Evaluating game syncs with the Game Sync Monitor involves the following tasks:

- [Adding Game Syncs to the Monitor](#)
- [Filtering Monitor Results](#)

Adding Game Syncs to the Monitor

The Game Sync Monitor can display data related to any RTPC being used on watched objects in your game. Before you can see the graphs of this data, however, you have to add the game syncs to be displayed in this view.

To add RTPCs to the Game Sync Monitor:

1. In the Game Syncs Watches pane of the Game Sync Monitor, click **Add**.

The Project Explorer - Browser dialog box opens.

2. Select the RTPC you want to add and click **OK**.

The RTPC is added to the Game Sync Monitor.

Related Topics

- [Evaluating Game Syncs with the Game Sync Monitor](#)
- [Filtering Monitor Results](#)

Filtering Monitor Results

When you add an RTPC to the Game Sync Monitor, all the watched game objects affected by that game sync are displayed by default in the graph view. Naturally, this can make for a crowded display when a game sync affects many objects. To simplify this view, you can unpin specific game object/game sync graphs to hide them.

To show or hide graphs in the Game Sync Monitor:

1. In the Game Sync Monitor, click the pin icon next to a game sync to display or hide its graph.

Related Topics

- [Evaluating Game Syncs with the Game Sync Monitor](#)
- [Adding Game Syncs to the Monitor](#)

Profiling Tips and Best Practices

Before using the Game Profiler in Wwise, you may want to review the following sections, which provide you with a series of tips and best practices that can help you profile and troubleshoot the different aspects of your game.

Game Profiler Performance

Because of the potentially large amount of data being transferred, capturing from the sound engine may affect Wwise's performance. Therefore, you may want to limit the type of information that is being generated by the Advanced Profiler by using the Profiler Settings dialog box. By deselecting one or more the information types, you can:

- Save network transfer bandwidth.
- Save memory space in Wwise.
- Save CPU time in game by not calculating the data.
- Save CPU time in Wwise by not processing or drawing the data.

Monitoring Performance While Capturing

If you want to monitor the performance while you are capturing data from the sound engine, you must select the Follow Capture Time option. If you don't select this option, after a short period of time, none of the views in the Profiler will seem like they are being updated and it will appear as if no information is being captured. If you forget to select the Follow Capture Time option, you can always scroll through the captured data by dragging the Performance Monitor time cursor (white) to a specific location on the graph. As you drag the time cursor, the other views in the Profiler will update automatically.

Troubleshooting Remote Connection Issues

- If you try to connect Wwise to your game, but the computer/console does not appear in the Remote Connections dialog box, verify the following:
 - The Communications module is properly initialized in your game. For more information, refer to the "Initializing Communications" section of the [Wwise SDK documentation](#).

- The same port number is specified for the "Game Discovery Broadcast Port" in your project settings and in the AkCommSettings structure used in your game to initialize communications. For more information, refer to [Specifying Network Ports](#) and the "Initializing Communications" section of the [Wwise SDK documentation](#).
- The computer/console is properly connected to the network. Note that some consoles have multiple network adapters, including one for debugging and one for regular network traffic. Make sure to connect all of them before trying to re-connect from Wwise.
- The computer/console is on the same subnet as the computer running Wwise.
- An active firewall is blocking the connection. If you have an active firewall on the computer running Wwise, try disabling it to see if the firewall is blocking the connection. If that works, try adding the Wwise application to the list of exceptions in your firewall (refer to your firewall's documentation for instructions). If you need to change the ports used for Wwise communication (including using fixed ports instead of dynamic ports), refer to [Specifying Network Ports](#) and the "Initializing Communications" section of the [Wwise SDK documentation](#).
- If you know the IP address of the remote computer/console, you can click "Connect to IP" in the Remote Connections dialog box and enter the IP address manually. If you can connect to the game in this manner, then it's only the broadcast from the remote computer/console that is blocked. After connecting a computer/console once, it will automatically appear in the "History" tab of the Remote Connections dialog box, so you won't have to enter the IP address when you try to reconnect in the future.
- The sound engine is in either debug or profile. You can't connect to a release build of the sound engine.
- If you try to connect to your game from a very large project and Wwise seems to hang on the connect screen, try the following:
 - Open your project, try emptying the current Soundcaster and Mixing Desk sessions, remove what is loaded into the transport control, and then try to connect to the game.

A little background information on why this may work. When Wwise connects to a game, it pushes information to the game, to make sure what is displayed in Wwise is in sync with what is in the game. But Wwise does not push everything because it would be too much and take too long. Instead, Wwise pushes only what is selected in the transport control and what is present in the active Soundcaster and Mixing Desk sessions. If you are trying to connect to your game from a Wwise project that has an active Soundcaster or Mixing Desk session containing a large amount of data, you may experience this problem.

Chapter 34. Managing SoundBanks

Overview	700
Understanding How SoundBanks are Loaded in a Game	702
Building SoundBanks	707
Managing SoundBanks	725
Defining Custom Attributes for Your SoundBanks	728
Generating SoundBanks for a Project	738
Using the CopyStreamedFiles Tool	743
Strategies for Managing SoundBanks	744
SoundBanks Tips and Best Practices	759

Overview

To effectively manage the audio and/or motion component of a game, Wwise puts all the audio and motion for your game into banks. A bank is basically a file that contains the audio and/or motion data, media, or both. These banks are loaded into a game's platform memory at a particular point in the game. By loading only what is necessary, you can optimize the amount of memory that is being used for audio and/or motion by each platform. Banks are the product of all your work and contain the final content that becomes part of your game.

In Wwise, there are two types of banks:

- **Initialization bank** - A special bank that contains all the general information about a project, including information on the bus hierarchy, and information on states, switches, and RTPCs. The Initialization bank is automatically created every time Wwise generates the SoundBanks. The Initialization bank is usually loaded once at the beginning of your game so that all the general project information is easily accessible during game play. It must be the first bank loaded when starting a game, or else the other banks may not load. Its file name is “Init.bnk”.
- **SoundBank** - A file that contains a combination of event data, sound, music, and motion structure data, and/or audio files. Unlike the Initialization bank, SoundBanks are generally loaded and unloaded at different points in the game to better utilize platform memory usage.

Because all platforms are different, Wwise allows you to easily tailor the SoundBanks for each platform and generate the SoundBanks for all platforms simultaneously. Wwise also provides you with tools for troubleshooting any issues related to your SoundBanks to make sure that you are respecting the limitations of the different platforms.

You must use only one Wwise project per game. If you have several people working on a very large project, you can divide up a project into separate work units. For information on using work units, refer to [Dividing Your Project into Work Units](#).

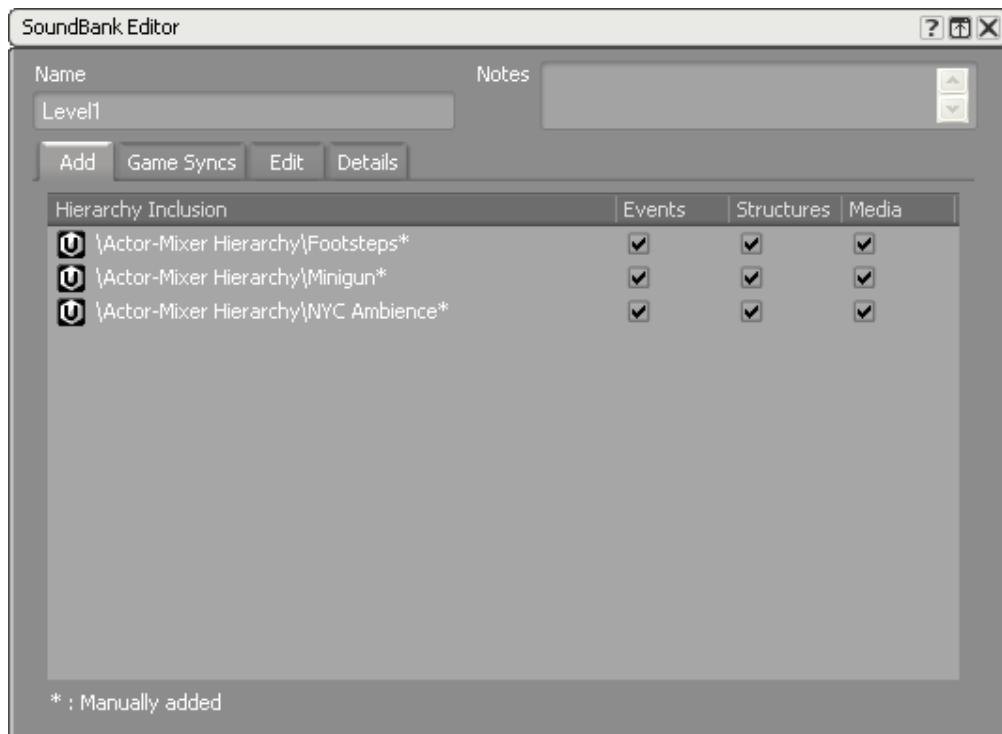
SoundBank Views in Wwise

To help you work more efficiently, a SoundBank layout is available in Wwise. This layout contains all the views you will need to create, manage, and generate the SoundBanks for your project, including the SoundBank Manager, SoundBank Editor, Project Explorer, and Event Viewer.

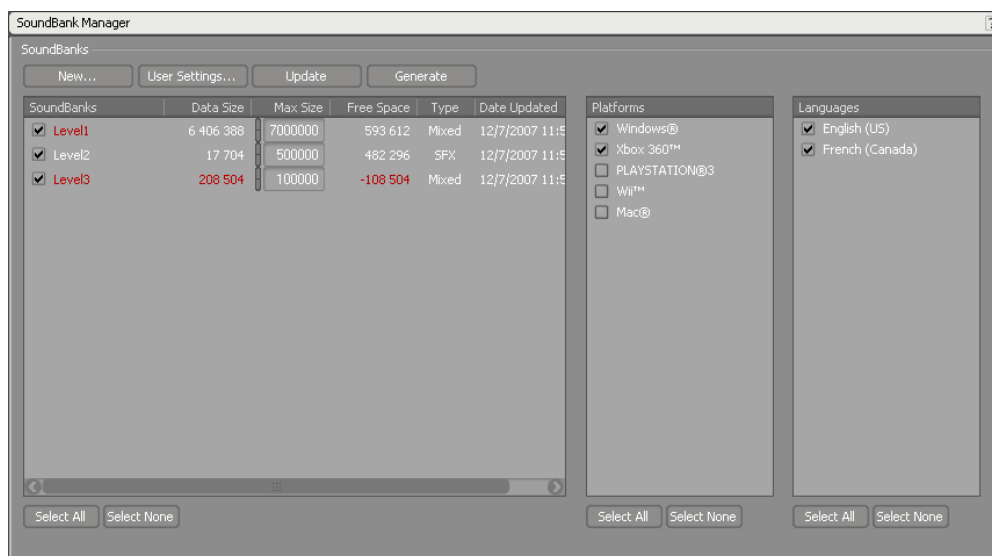
The SoundBank Editor is where you build your SoundBanks by adding different events, sound, music, and motion structures, and audio files to the SoundBank. You can open the SoundBank Editor by double-clicking a SoundBank in either the Project Explorer or SoundBank Manager.

The SoundBank Editor consists of the following four tabs:

- **Add** - Displays only the actual events, object hierarchies, work units, and folders, that were added to the SoundBank. Any corresponding child objects that are also automatically added to the SoundBank are only displayed on the Edit tab. On the Add tab, you determine what types of information and/or media will be included in the SoundBank per hierarchical element.
- **Game Syncs** - Displays a list of states, switches, and triggers related to the events and sound structures that have been included on the Add tab. On this tab, you can exclude specific game syncs from a SoundBank. By excluding a game sync, you also exclude any related sound structures and media files.
- **Edit** - Displays a detailed list of each individual event, object structure, and media file, including all child objects, that are associated with the project elements on the Add tab. Additional information is displayed for the media files in the SoundBank, including the sample rate, audio format, and file size. By having this information available on this tab, you can easily tweak the conversion settings of each individual file to meet the constraints or limitations of a particular platform. You can filter the list by language and object type and then deselect any project elements that you want to exclude from the SoundBank.
- **Details** - Displays detailed information about the size of the different elements within the selected SoundBank as well as reporting any files that may be missing or replaced.



At any point in the development of your project, you can generate your SoundBanks from the SoundBank Manager. The SoundBank Manager displays a list of SoundBanks that have been created along with some basic information about the type and size of each SoundBank. It also includes a separate list of platforms and languages that, when selected, determines which SoundBanks will be generated.



Understanding How SoundBanks are Loaded in a Game

Before learning about how SoundBanks are built and generated in Wwise, it is important to understand the different ways that SoundBank information can be loaded and managed by your game. The method you chose will depend on a number of different factors including, the type of game you are developing, the platforms on which it will run, and the constraints set by the project team itself.

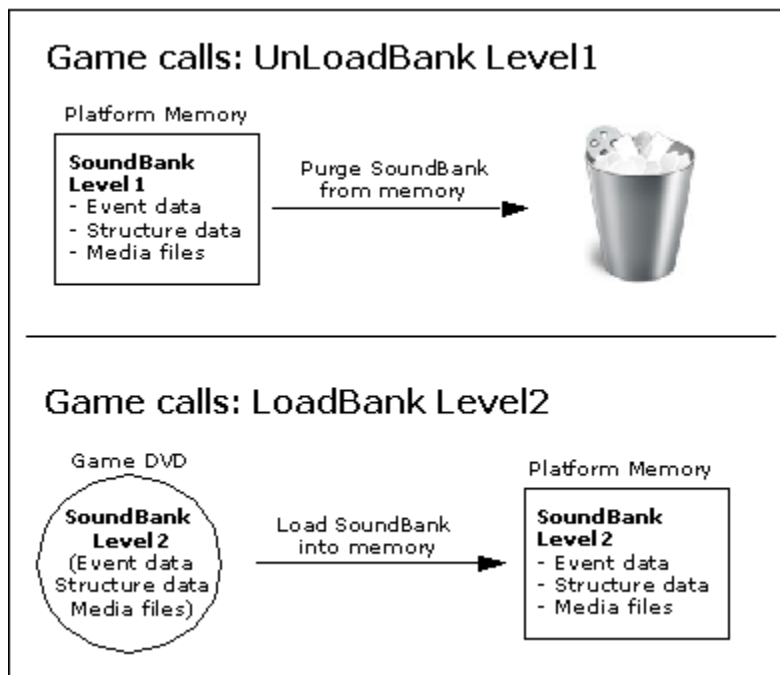
In an attempt to be as flexible as possible and to meet the requirements of almost any type of game, Wwise offers several different methods for loading audio and motion in a game, including the following:

- [Loading a Bank](#)
- [Preparing a Bank \(All Content\)](#)
- [Preparing Action Events](#)

Loading a Bank

The first method uses SoundBanks that contain a mix of event data, object structure data, and media. The entire contents of these SoundBanks are loaded and unloaded at particular points in the game to ensure that event data and related media are ready to be played when triggered.

The following illustration demonstrates how SoundBanks created using the traditional method are loaded and unloaded from platform memory as the player moves from Level 1 to the Level 2 in the game.



Since all the data and media for a particular SoundBank are loaded into memory at the same time, this method not only ensures that all data and media are ready to play when required, but it performs very little disk seeking during gameplay, which frees up the disk for other disk intensive tasks.

The main drawback with this method is that a substantial amount of memory is taken for the entire time that the SoundBanks are loaded, which gives you less flexibility when dealing with large, sophisticated games. This method also explicitly loads all the content within a SoundBank without verifying to see if media files are already loaded into memory. This may cause the same media file to be loaded into memory more than once. Despite these drawbacks, the traditional method can still be useful in many situations, for example, a pinball game, where all data and media must be available at all times.

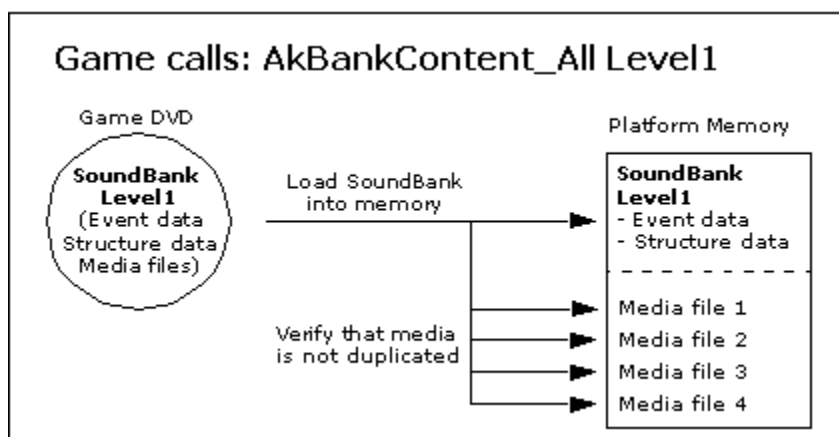
Preparing a Bank (All Content)

To overcome some of the drawbacks of the LoadBank mechanism, you can prepare your banks instead of loading them using `AkBankContent_All`. When using this method, your banks may still include all content types (events, structure data, and media files), but instead of loading the media files outright, this method loads all media into memory by using the prepare event mechanism. By using this mechanism to load media, Wwise first checks to see

if a media file already exists in memory before loading it. This keeps memory usage at a minimum by avoiding any duplication of media files in memory.

Along with memory savings, this method also guarantees that disk access will be sequential. This avoids the random disk seeking that is likely to occur when events are prepared one at a time using `PrepareEvent`.

The following illustration demonstrates how the “Prepare Bank (All Content)” mechanism loads metadata and content into your platform's memory.



Preparing Action Events

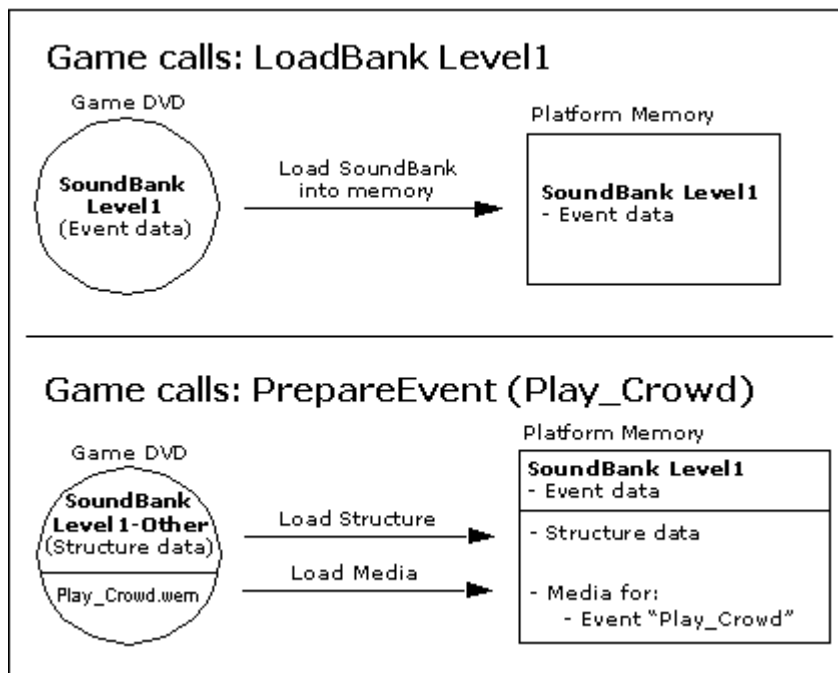
The “`PrepareEvent`” method dynamically loads the media only when it is absolutely required. With this method, the action event metadata must be in a bank and the associated media files must be accessible in the file system. The corresponding structure metadata can be included in the same bank as the events, or in a separate bank. When using this method, the bank that contains the action event metadata is loaded and kept in memory using `LoadBank`. Prior to action events being called by the game, they are “prepared” by the sound engine. Preparing an event will load all referenced media files from the file system, and also load all referenced structure metadata from a bank, if it is not already loaded. When the action event is no longer required, it can be “unprepared” and the corresponding media files are purged from memory. To keep memory usage at a minimum and to avoid any duplication of media files in memory, Wwise performs a check before loading any media file to ensure that the media file is not already in memory.



Note

Only action events can be prepared in advance. The “`PrepareEvent`” method does not work with dialogue events.

The following illustration shows how events can be prepared in advance so that only required media files are loaded into memory.

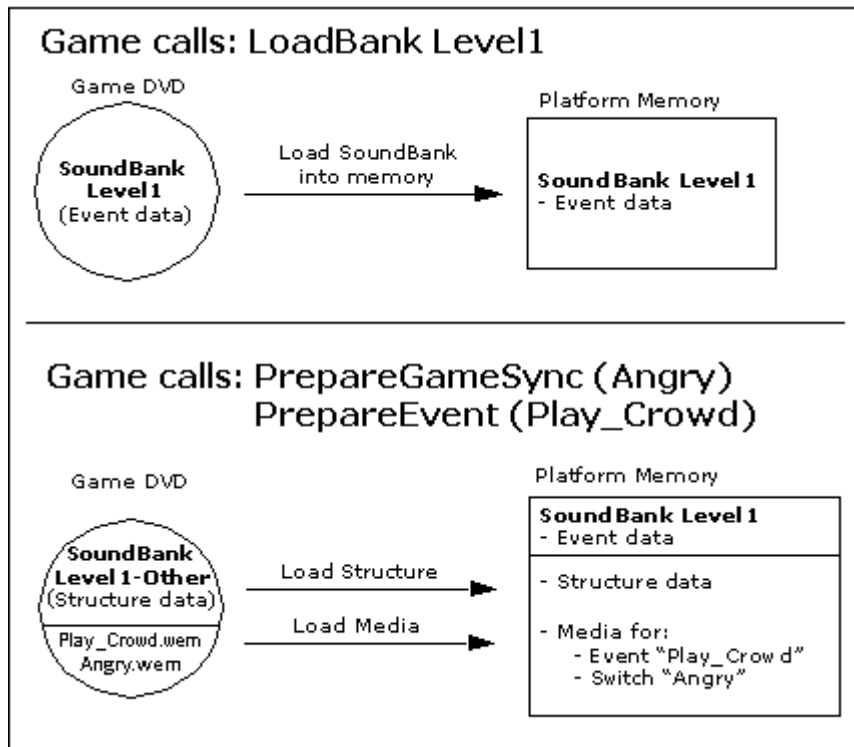


When the metadata (also referred to as structures) is not stored in the same bank as the events, Wwise needs a way to find the corresponding data that resides in another bank. To do this, Wwise includes references to the corresponding content stored in other SoundBanks. Wwise can use either names or IDs to refer to other SoundBanks. To use the SoundBank name in the sound engine, you must select the Use SoundBank Names option on the SoundBanks tab of the Project Settings dialog box. To use IDs, deselect that option.

Media that is referenced by a bank must be stored as loose files on disk, or otherwise be resolvable by the low level IO (such as in a File Package).

This method is generally very efficient in terms of memory usage, but it does require additional disk seeking, which may not be appropriate in situations where many files are already being streamed from the disk. Also, in cases where the game uses states and switches, media files could be loaded into memory unnecessarily. For example, if you have different crowd sounds for the different moods or energy level of the crowd within your game, you wouldn't want to load all of them into memory if only the angry switch sounds are valid within a particular zone of the game. To overcome this problem, you can also "prepare" specific states and/or switches that load only those media files that are associated with the prepared states or switches.

The following illustration shows how switches can be prepared in advance to further limit the amount of media that is loaded into memory at any particular point in the game.



Although preparing game syncs in advance can optimize memory usage, be aware that it also reduces the speed at which media is loaded into memory. Longer read times result from the sound engine needing to seek through the disk to find the sounds that correspond to the prepared game syncs.

By dynamically loading only those audio files that are required, this second method gives you greater flexibility to handle situations where you have either very large zones or levels with many sounds or very limited amounts of memory available to store both event data, structure data, and media.

As you can see, each method has their own strengths and weaknesses. The method you choose will depend on the specific scenarios, requirements, and limitations of your game. After deciding on a strategy, whether it be a single method, or a combination of several methods, you can begin to populate and fine-tune your SoundBanks accordingly.



Note

The LoadBank, AkBankContent_All, PrepareEvent, PrepareGameSync, and AkBankContent_StructureOnly

functions are accessible through the Wwise API. For more information on loading banks and preparing events and/or game syncs, refer to the Integrating Banks section of the [Wwise SDK documentation](#).

For a detailed overview of the different methods you can use to manage your SoundBanks, refer to [Strategies for Managing SoundBanks](#).

Building SoundBanks

When determining how many SoundBanks to create, you and your programmers would typically parse the whole game to get a list of all the Wwise events that are integrated in the game. From there, you can define the size limit and number of SoundBanks that will be required for your game. Then you can organize the events, object structures, and media into the various SoundBanks based on the method chosen as well as the characters, objects, zones, or levels in game.

Wwise does not restrict the number of SoundBanks that can be loaded into memory. There is also no restriction on the size of the SoundBank itself. Each project will be different, and it is up to you to decide what works best for your project.

The following tasks are involved in building a SoundBank:

- [Creating a SoundBank](#)
- [Populating a SoundBank](#)
- [Managing the Contents of a SoundBank](#)
- [Generating SoundBanks for a Project](#)



Note

The only restriction to the size of a SoundBank is the memory available on the system or console.

Creating a SoundBank

You can create individual SoundBanks manually in Wwise or you can create a batch of SoundBanks by importing a definition file that was generated from the 3D application or level editor that you are currently using to integrate events into your game. For more information on importing definition files, refer to [Populating a SoundBank by Importing a Definition File](#).

You can manually create SoundBanks from two different areas of Wwise:

- [To create a SoundBank from the Project Explorer:](#)

- [To create a SoundBank from the SoundBank Manager:](#)



Note

When you first create a SoundBank, Wwise automatically names it “New_SoundBank_X”. You can rename the SoundBank to give it a more meaningful name. If you plan to rename a SoundBank, you should do it early in the development process because making a change later on may require additional programming.

To create a SoundBank from the Project Explorer:

1. Switch to the SoundBank layout by doing one of the following:

From the menu bar, click **Layouts > SoundBank**.

Press **F7**.

2. In the Project Explorer, switch to the **SoundBanks** tab.
3. Do one of the following:

Select a virtual folder or work unit and click the **SoundBank** icon in the Project Explorer toolbar.

Right-click a virtual folder or work and select **New Child > SoundBank** from the shortcut menu.

A new SoundBank is highlighted in the Project Explorer.

4. Type the new name of the SoundBank and press **Enter**.

The SoundBank is created and added to the SoundBank list.



Note

SoundBank names can contain only letters, numbers, and underscores. They must also start with either a letter or underscore.

5. In the Max Size text box, specify the maximum amount of in-game memory that you want to allocate for the SoundBank.

To create a SoundBank from the SoundBank Manager:

1. Switch to the SoundBank layout by doing one of the following:

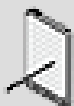
From the menu bar, click **Layouts > SoundBank**.

Press F7.

2. In the SoundBank Manager, click the **New** button.

The New SoundBank dialog box opens.

3. In the SoundBanks hierarchy, select the work unit into which you want to create the SoundBank.
4. In the name field, replace the default name with a suitable name for the new SoundBank.



Note

SoundBank names can contain only letters, numbers, and underscores. They must also start with either a letter or underscore.

5. Click **OK**.

The SoundBank is created and added to the SoundBank list.

6. In the Max Size text box, specify the maximum amount of in-game memory that you want to allocate for the SoundBank.



Note

If you are using a workgroup plug-in and the work unit in which you created the new SoundBank is not checked out, Wwise will ask you if you want to check out the work unit only when you try to save the project.

Related Topics

- [Populating a SoundBank](#)
- [Managing the Contents of a SoundBank](#)
- [Monitoring the Details of a SoundBank](#)
- [Generating SoundBanks for a Project](#)

Populating a SoundBank

Depending on the type of game you are creating as well as the SoundBank strategy you have chosen, your SoundBanks may contain different types of information and media. The strategy you choose will ultimately decide the size and contents of your SoundBanks. A SoundBank can include a combination of any of the following types of project elements:

- Event data

- Sound, music, or motion structure data
- Media files

You can populate a SoundBank using either of the following methods:

- [Populating a SoundBank by Importing a Definition File](#) - by importing a definition file.
- [Populating a SoundBank Manually](#) - by dragging project elements, such as events, object structures, and work units, from the Project Explorer into the selected SoundBank.

Populating a SoundBank by Importing a Definition File

One of the methods for creating and populating SoundBanks is by importing a definition file that was generated from the 3D application or level editor that you are currently using to integrate events into your game. A definition file is a tab delimited text file that lists all the events in your game, classified by SoundBank. The definition file must include the name of the SoundBank and the corresponding event name, separated by a tab. The following example demonstrates how a definition file should be written so that it can be read by Wwise.

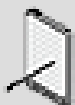
Using Keywords to Include/Exclude Elements from a SoundBank

The definition file can also include special keywords that define the types of project elements that will be included or excluded from the SoundBanks. The following keywords can be used:

- **Event** - Specifies that the SoundBank will include event information.
- **Structure** - Specifies that the SoundBank will include sound, music, and/or motion structure information.
- **Media** - Specifies that the SoundBank will include media files.
- **-GameSyncExclusion** - Specifies that a particular game sync will be excluded from a SoundBank. When you exclude a game sync, all related sound structures and media files are excluded as well. The **-GameSyncExclusion** keyword must be used in combination with one of the following keywords:
 - **State** - Specifies that a particular state and all its related objects and media files will be excluded from the SoundBank. This keyword must be followed by the state group name and the state name, separated by tabs.
 - **Switch** - Specifies that a particular switch and all its related objects and media files will be excluded from the SoundBank. This keyword must be followed by the switch group name and the switch name, separated by tabs.
 - **Trigger** - Specifies that a particular trigger and all its related objects and media files will be excluded from the SoundBank. This keyword must be followed by the name of the trigger.

- **-DialogueEvent** - Specifies that a particular dialogue event will be included in the SoundBank. This keyword must be followed by the name, GUID, or short ID of the dialogue event. It can also be followed by the **Event**, **Structure** and/or **Media** keywords as inclusion options.
- **-EffectShareset** - Specifies that a particular effect shareset will be included in the SoundBank. This keyword must be followed by the name, GUID, or short ID of the effect shareset. It can also be followed by the **Structure** and/or **Media** keywords as inclusion options.
- **-AuxBus** - Specifies that a particular Aux Bus will be included in the SoundBank. This keyword must be followed by the name, GUID, or short ID of the bus. It can also be followed by the **Structure** and/or **Media** keywords as inclusion options.

In the previous illustration, the first SoundBank called “My_SoundBank_Normal” will include all event data, structure data, and media files associated with Event_01. The second SoundBank called “My_SoundBank_EventandStructure” will include only the event and structure data associated with Event_02. The third and final SoundBank called “My_SoundBank_Media” will include only the media files associated with Event_03.



Note

All events included in the definition file must already be created in Wwise. If any are missing, they will appear as “Event Missing” in the Import Definition Log.

When using the **-GameSyncExclusion** keyword, you must create a separate entry on a new line for each exclusion. By excluding a game sync, you exclude all corresponding object structures and media files as well. The following illustration demonstrates how the **-GameSyncExclusion** keywords should be written within a definition file. (→ indicates the TAB character)



Note

The names of the game syncs (state groups, states, switch groups, switches, and triggers) do not require the use of quotes (“”).

Using IDs Instead of Strings to Identify Events and Effects

If your game is not using strings for event names, you can use either of the following systems to define the events in the definition file:

- Hexadecimal
- Decimal



Note

The hexadecimal and decimal systems can be used to identify events within a definition file, but not states, switches, and triggers in game sync exclusions.

The following illustration demonstrates how to define events in a definition file using the three different systems. (→ indicates the TAB character)



Tip

The SoundBank definition file can also be used to track which events have already been integrated into your game, which ones are missing, and which ones still need to be created in Wwise. One of the programmers can generate the list of events from the game and then you can import the definition file into Wwise. You can use the information in the log file to match up the events in game with those created in Wwise.

To create a SoundBank by importing a definition file:

1. In the Project Explorer, switch to the SoundBanks tab.
2. Right-click the work unit into which you want to create the SoundBanks.
3. From the shortcut menu, select **Import SoundBank Definition**.

The Open dialog box opens.

4. Navigate to the location where the definition file is saved.
5. Click **Open**.

The Import Definition Log dialog box opens.

6. Review the import activity in the log. The import activity can be any one of the following:

Inclusion Added - A new event or effect has been added to the existing SoundBank.

SoundBank Created - A new SoundBank has been created.

Inclusion Removed - An event or effect has been removed from the existing SoundBank.

Event Missing - An event no longer exists in the project or has yet to be created.

Effect Missing - An event no longer exists in the project or has yet to be created.

Exclusion Added - A game sync was excluded from the existing SoundBank.

Exclusion Deleted - A game sync was re-included into an existing SoundBank.

Exclusion Missing - A game sync no longer exists in the project or has yet to be created, so the exclusion could not be added or deleted from the exclusion list.

No Change Detected - The imported SoundBank is identical to the one already in Wwise.

7. Click Close.

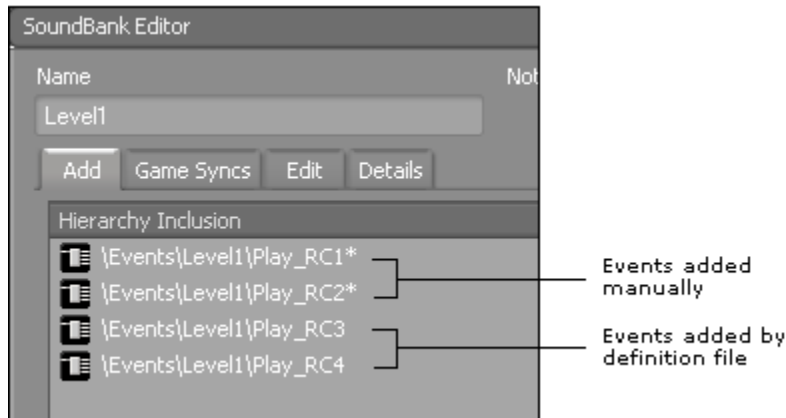
The SoundBanks that are defined in the definition file are created in Wwise and populated with the specified events, object structures, and media.

Related Topics

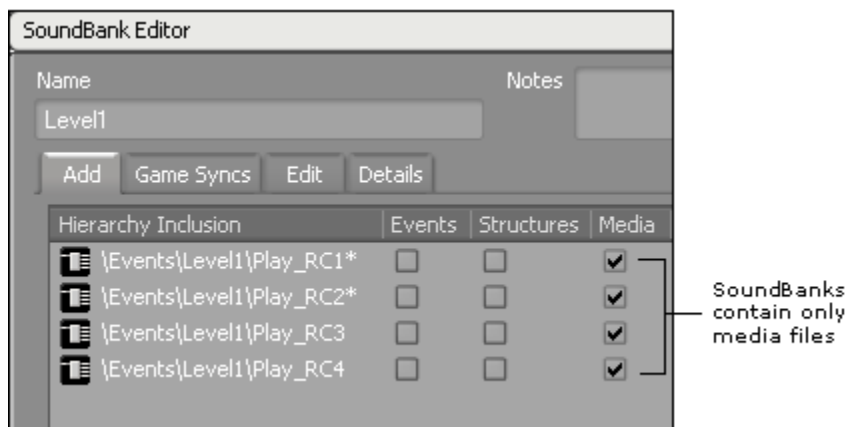
- [Creating a SoundBank](#)
- [Including/Excluding Project Elements from a SoundBank](#)
- [Including/Excluding Project Elements Using Game Syncs](#)
- [Searching for Elements within a SoundBank](#)
- [Filtering the List of Elements within Your SoundBank](#)
- [Removing Project Elements from a SoundBank](#)
- [Monitoring the Details of a SoundBank](#)
- [Generating SoundBanks for a Project](#)

Populating a SoundBank Manually

After you have created all your SoundBanks by importing a definition file, you can then fine-tune their contents manually adding and excluding individual events, game syncs, and sound, music, or motion objects. To help you identify which project elements have been added or excluded manually from a SoundBank, an asterisk (*) is added next to the entry on the Add and Game Syncs tabs of the SoundBank Editor.



When you add a project element to a SoundBank, all corresponding events, object structures, and media files are automatically added to the SoundBank as well. You can, however, modify the contents of the SoundBank by including only those types of project elements that you want. For example, you may want your SoundBank to include only media.



To help you work more efficiently, Wwise allows you to drag complete structures, work units, and folders, from the Project Explorer into the SoundBank Editor. If these project elements contain child objects, they are automatically added to the SoundBank as well, although they are only displayed on the Edit tab.

When a parent object, work unit, or folder is added to a SoundBank, an active link to the original project element is maintained. For example, let's say you have a SoundBank that contains one event work unit. When you subsequently add two events to the work unit, they are automatically added to the SoundBank. If you load the SoundBank into the SoundBank Editor, you will notice that the Add tab still only displays the original event work unit, but the Edit tab now displays all the events within the work unit, including the two that were just added. By maintaining this link, Wwise ensures that your SoundBanks always include the latest modifications made to your project.



Note

If a SoundBank contains a project element that has been unloaded from the project, the project element will appear in yellow on the Add tab of the SoundBank Editor.

To populate a SoundBank manually:

1. Switch to the SoundBanks layout by doing one of the following:

From the menu bar, click **Layouts > SoundBank**.

Press **F7**.

2. Double-click one of the SoundBanks in the SoundBank Manager to load it into the SoundBank Editor.
3. Drag any of the following elements from the Project Explorer to the Add tab of the SoundBank Editor:

Actor-Mixers, Containers, Sounds, Motion FX, and Music objects

Folders

Events

Work units

The whole hierarchy of project elements and all associated events, object structures, and media files are automatically added to the SoundBank. You are now ready to further refine the contents of your SoundBank by deciding which types of project elements you want to include.

Related Topics

- [Creating a SoundBank](#)
- [Including/Excluding Project Elements from a SoundBank](#)
- [Including/Excluding Project Elements Using Game Syncs](#)
- [Searching for Elements within a SoundBank](#)
- [Filtering the List of Elements within Your SoundBank](#)
- [Removing Project Elements from a SoundBank](#)
- [Monitoring the Details of a SoundBank](#)
- [Generating SoundBanks for a Project](#)

Managing the Contents of a SoundBank

After you have created and populated your SoundBanks, you will most likely need to fine-tune the contents by including, excluding, and removing particular

elements from the SoundBank. Since a SoundBank can contain many different project elements, Wwise provides you with different methods for including and excluding project elements, as well as a search tool and two separate filtering options to help you locate objects quickly.

When managing the contents of a SoundBank, you can carry out the following tasks:

- [Including/Excluding Project Elements from a SoundBank](#)
- [Including/Excluding Project Elements Using Game Syncs](#)
- [Searching for Elements within a SoundBank](#)
- [Filtering the List of Elements within Your SoundBank](#)
- [Moving Project Elements between SoundBanks](#)
- [Removing Project Elements from a SoundBank](#)

Including/Excluding Project Elements from a SoundBank

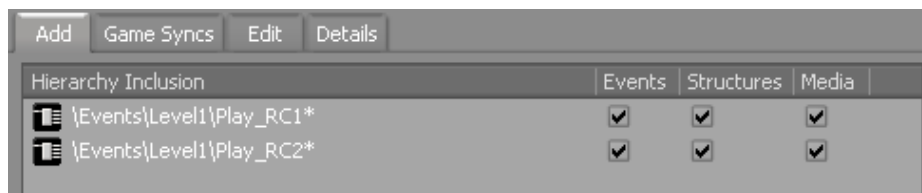
When you add an event or sound structure to a SoundBank, Wwise automatically includes all corresponding events, sound structures, and audio files. Depending on the SoundBank strategy you are using for your game, you may want to remove all the events, sound structures, or media files from the SoundBank.

Example - Including/Excluding Elements from a SoundBank

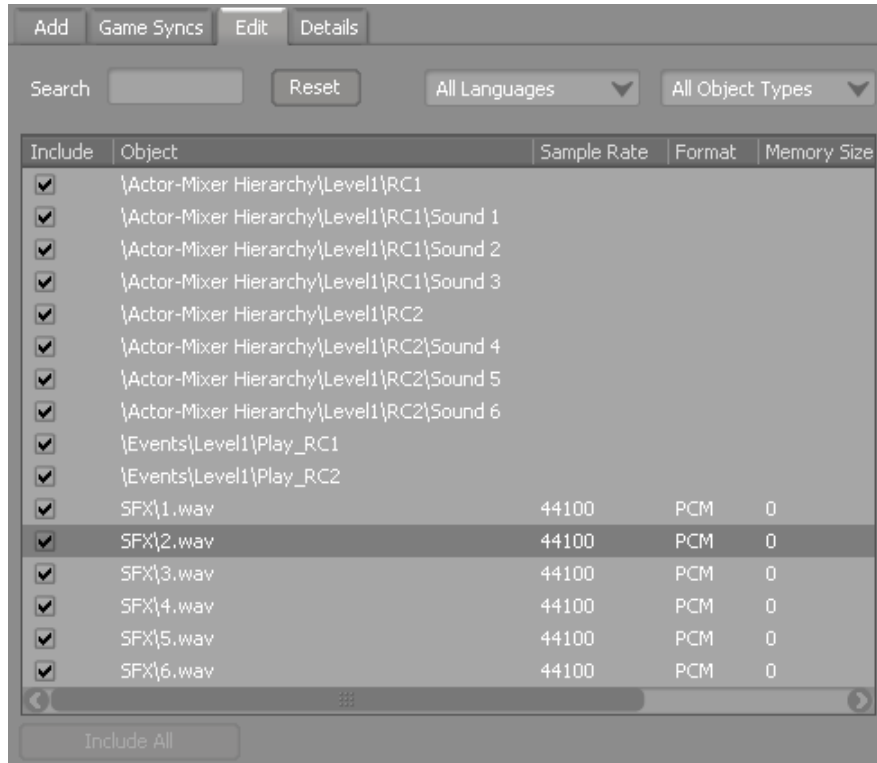
Let's say you have a small project with the following project elements:

- Two random containers
- Six sound objects
- Six audio files
- Two play events

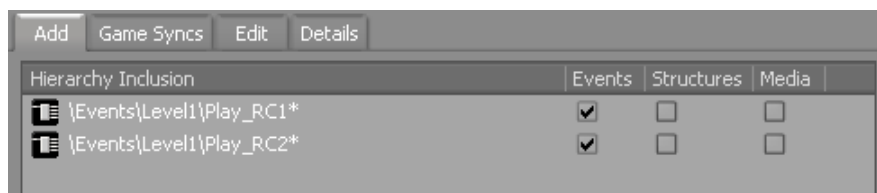
You want to build one SoundBank for all the events in your project because you plan to load media files dynamically by preparing events in advance of them being called. In this case, you could create a SoundBank and then drag the two events onto the Add tab of the SoundBank Editor.



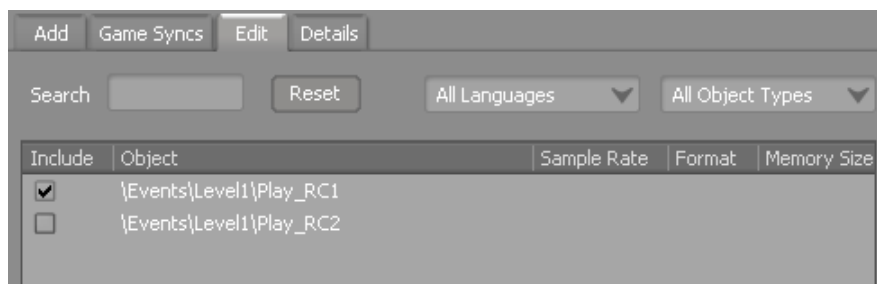
By default, the data for the two random containers and the six sound objects, as well as the six audio files, will all be added to the SoundBank as well. You can view the entire contents of your SoundBank on the Edit tab of the SoundBank Editor.



To include only the events in this SoundBank, you can return to the Add tab and clear the check boxes under the Structures and Media columns.



Now your SoundBanks only include the data for the two events. You can fine-tune your SoundBank even further, by excluding individual elements on the Edit tab of the SoundBank Editor. For example, you may only want one of the events included in this SoundBank, so you could exclude the one you don't want.



For SoundBanks that do include media, you can view specific information about each media file, including its sample rate, audio format, and file size. By having this additional information, you can easily tweak the conversion settings of each individual file to respect the limitations of a particular platform. To change the conversion settings of a media file, simply right-click the entry in the list and select Conversion Settings.

By adopting the right strategy for your game and by having the flexibility to include and exclude individual elements from your SoundBank as well as modify the conversion settings of individual media files, you can effectively overcome the memory constraints of your game.

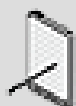
To include project element types in a SoundBank:

1. Load a SoundBank into the SoundBank Editor.
2. For each entry in the list, select any or all of the following project element types to include them in your SoundBank:

Events - Includes all corresponding event data in the SoundBank.

Structures - Includes all corresponding object structure data in the SoundBank.

Media - Includes all corresponding media files in the SoundBank.



Note

To exclude all events, object structures, or media files associated with a particular project element, deselect the corresponding check box. To help speed up the process, you can multi-select entries in the Hierarchy Inclusion list, and then deselect one of the check boxes. This will exclude the type for all selected entries.

To exclude individual project elements from a SoundBank:

1. Load a SoundBank into the SoundBank Editor.
2. Switch to the **Edit** tab.

A complete list of all events, object structures, and media files included in the SoundBank is displayed.

3. Locate the project element you want to exclude from the SoundBank using the Search or filter tools.
4. Deselect the corresponding check box for the event, object structure object, or media file.

The project element and all associated project elements are now excluded from the SoundBank.



Note

When a parent object is excluded from a SoundBank, all corresponding child objects get excluded as well. You will not be able to select any of the child objects until the parent object has been added back to the SoundBank.

5. Repeat steps 3 and 4 until all project elements that you want removed have been excluded from the SoundBank.

Related Topics

- [Including/Excluding Project Elements Using Game Syncs](#)
- [Searching for Elements within a SoundBank](#)
- [Filtering the List of Elements within Your SoundBank](#)
- [Moving Project Elements between SoundBanks](#)
- [Removing Project Elements from a SoundBank](#)

Including/Excluding Project Elements Using Game Syncs

Another quick and easy way to include or exclude sounds from a SoundBank is to use game syncs. When you add events or sound structures to a SoundBank, a list of related game syncs is automatically created on the Game Syncs tab of the SoundBank Editor. You can remove certain elements from your SoundBank by excluding specific game syncs. When a game sync is excluded from a SoundBank, all object structures and media files that reference that game sync will be excluded as well.

By including/excluding sounds based on their relationship with game syncs, you have more control over what sounds are loaded at specific points in the game. This means you can better manage the memory limitations of the game's platform.



Note

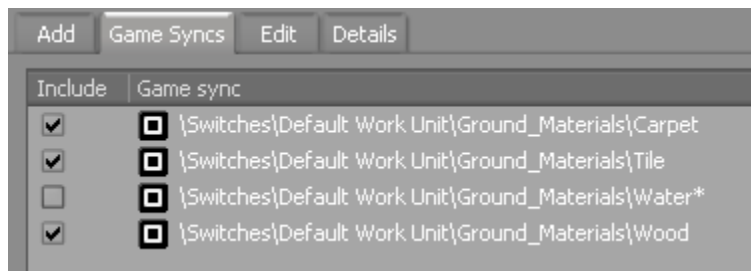
Game parameters can't be used to exclude project elements from a SoundBank and are therefore not listed on the Game Syncs tab of the SoundBank Editor.

Example - Including/Excluding Elements Using Game Syncs

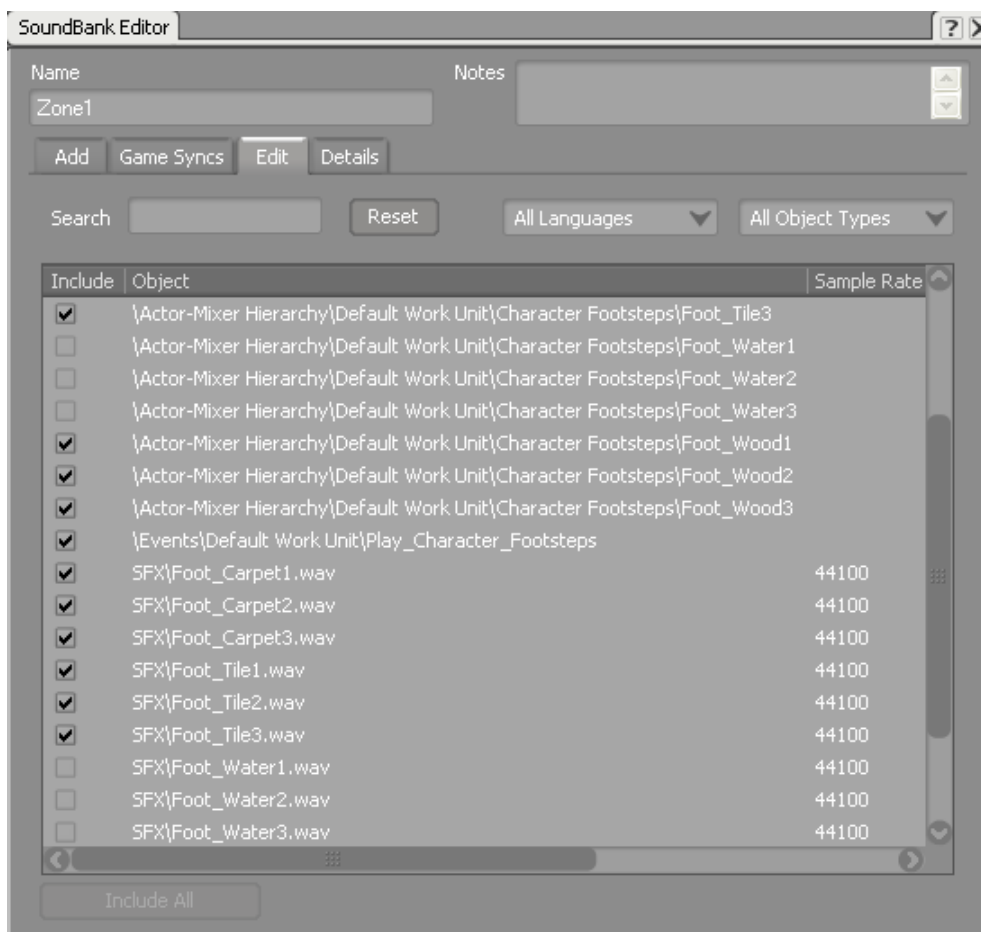
Let's say you have a small game made up of three different zones. To complete the game, the main character must walk through zones 1, 2, and 3. In Wwise, you are using a switch container to handle the footsteps sounds of your main character. You have created the following four switches for the different ground materials on which the character can walk:

- Carpet
- Tile
- Wood
- Water

In your game, the water switch is only used in Zone 2 and 3, whereas the other three switches are used in all three zones. You decide to create three SoundBanks; one for each zone. Since the switch container must be included in each SoundBank, you will want to remove the “water-related” events, sound structures, and media files from the SoundBank for Zone 1. To do so, you could load the SoundBank for Zone1, switch to the Game Syncs tab, and then deselect the “Water” switch.



This will exclude “water” related sound structures and media files from the SoundBank “Zone1”. If you switch to the Edit tab, you will notice that the sound structures, events, and media files that reference this switch will no longer be included in the SoundBank.



Game syncs can also be excluded from a SoundBank automatically by importing a definition file that includes a game sync exclusion. For more information on using definition files, refer to [Populating a SoundBank by Importing a Definition File](#).



Note

You may notice that some project elements and media files on the Edit tab are grayed out even before you've started to manually exclude them. This can occur when a switch container is added to the SoundBank, and includes objects that have not yet been assigned to a switch. Since these objects will never be played in game, they are automatically excluded from the SoundBank.

To exclude project elements from a SoundBank using game syncs:

1. Load a SoundBank into the SoundBank Editor.
2. Switch to the **Game Syncs** tab.

A complete list of game syncs related to the events and/or objects on the Add tab is displayed.

3. Locate the game sync you want to exclude from the SoundBank and deselect the corresponding check box.

The game sync and all related objects and media files are now excluded from the SoundBank.

4. Repeat step 3 until all game syncs that you want removed have been excluded from the SoundBank.

Related Topics

- [Including/Excluding Project Elements from a SoundBank](#)
- [Searching for Elements within a SoundBank](#)
- [Filtering the List of Elements within Your SoundBank](#)
- [Moving Project Elements between SoundBanks](#)
- [Removing Project Elements from a SoundBank](#)

Including Plug-in Media in a SoundBank

Some effects, such as the Convolution Reverb, require media files to work. For example, the Convolution Reverb requires an impulse response file as its plug-in media. Those plug-in media files need to be included inside a SoundBank for the effect to function normally in the game. You also need to make sure the SoundBanks containing the media files are loaded at the moment you need the effect to be instantiated.

There are multiple ways of including the plug-in media files in a SoundBank:

- You can insert the Shareset effect object inside the inclusion list of a SoundBank.
- You can insert the owner object (bus, auxiliary bus or Actor-Mixer Hierarchy object) inside the inclusion list of a SoundBank.
- You can insert the event inside the inclusion list of a SoundBank, if the effect is owned by an Actor-Mixer Hierarchy object.

It is possible to verify that the plug-in media files were included in a SoundBank by looking at the SoundBank Editor/Edit tab.

Searching for Elements within a SoundBank

Depending on the complexity of your game, you may have hundreds, if not thousands of different project elements in a SoundBank. It is essential, therefore, that you have the necessary tools to find specific project elements quickly. To help you find elements on the Edit tab, a search tool is available.

To search for an element within a SoundBank:

1. Load a SoundBank into the SoundBank Editor.
2. Switch to the **Edit** tab.
3. In the Search field, type the name of the project element you want to find.

Wwise filters the list to include only those objects that match the name you have entered.

4. Click **Reset** to erase the Search field and to display the entire contents of the SoundBank again.

Related Topics

- [Including/Excluding Project Elements from a SoundBank](#)
- [Including/Excluding Project Elements Using Game Syncs](#)
- [Filtering the List of Elements within Your SoundBank](#)
- [Moving Project Elements between SoundBanks](#)
- [Removing Project Elements from a SoundBank](#)

Filtering the List of Elements within Your SoundBank

To help you find specific elements within your SoundBank, you can filter the list by language or by object type.

To filter the list of elements within your SoundBank:

1. Load a SoundBank into the SoundBank Editor.
2. Switch to the **Edit** tab.
3. To filter the list elements by language, select one of the following options from the Language Filter:

All languages - Displays project elements for all languages in your project

“Language” - Displays project elements for the selected language only.

4. To filter the list of elements by object type, select one of the following options from the Object Type list:

All object types - Displays all objects in the SoundBank

Events - Displays only the events in the SoundBank

Structures - Displays only the object structures in the SoundBank.

Media - Displays only the media files in the SoundBank.

Related Topics

- [Including/Excluding Project Elements from a SoundBank](#)

- [Including/Excluding Project Elements Using Game Syncs](#)
- [Searching for Elements within a SoundBank](#)
- [Moving Project Elements between SoundBanks](#)
- [Removing Project Elements from a SoundBank](#)

Moving Project Elements between SoundBanks

Over the course of a typical game development project, the number and contents of your SoundBanks will change. It is important, therefore, that you be able manage these SoundBanks efficiently. To help you better manage the contents of your SoundBanks, Wwise allows you to easily move project elements from one SoundBank to another.

To move project elements between SoundBanks:

1. Load a SoundBank into the SoundBank Editor.
2. From the Add tab of the SoundBank Editor, select one or more project elements that you want to move to another SoundBank.
3. Drag the selection from the Add tab of the SoundBank Editor to a different SoundBank in the SoundBank Manager.

The selected project elements are removed from the original SoundBank and added to the new SoundBank.

Related Topics

- [Including/Excluding Project Elements from a SoundBank](#)
- [Including/Excluding Project Elements Using Game Syncs](#)
- [Searching for Elements within a SoundBank](#)
- [Filtering the List of Elements within Your SoundBank](#)
- [Removing Project Elements from a SoundBank](#)

Removing Project Elements from a SoundBank

When you remove an element from a SoundBank, all corresponding events, object structures, and media files are also removed.

Although invalid project elements are ignored by Wwise during the generation process and will not cause errors or take up additional space, you should remove them from your SoundBanks in order to maintain the integrity of your project. Invalid events and object structures appear on the Add tab of the SoundBank Editor with the word “Missing” after their name.

To remove a project element from a SoundBank:

1. Load a SoundBank into the SoundBank Editor.
2. On the Add tab, select one or more objects that you want to remove.

3. Press the **Delete** key.

The project elements are removed from the SoundBank along with any corresponding events, object structures, and/or media files.



Note

If an object or media file is referenced by more than one project element in the SoundBank, removing one of the project elements will not automatically remove the object or media file because it is still being referenced by another project element.

Related Topics

- [Searching for Elements within a SoundBank](#)
- [Including/Excluding Project Elements from a SoundBank](#)
- [Including/Excluding Project Elements Using Game Syncs](#)
- [Filtering the List of Elements within Your SoundBank](#)
- [Moving Project Elements between SoundBanks](#)

Managing SoundBanks

After your initial SoundBanks are created, you can continue to make changes throughout the development process. When managing SoundBanks, you can carry out the following tasks:

- [Renaming a SoundBank](#)
- [Deleting SoundBanks](#)
- [Monitoring the Details of a SoundBank](#)

Renaming a SoundBank

After you have created a SoundBank, you can go back and rename it as needed.



Note

If you plan to rename a SoundBank, you should do it early in the development process because making a change later on may require additional programming.

To rename a SoundBank:

1. In the Project Explorer, switch to the **SoundBanks** tab.

2. Select the SoundBank you want to rename and then do one of the following:

Press F2.

Click the SoundBank name field.

The name of the SoundBank becomes highlighted.

3. Replace the SoundBank name with the new name and press **Enter**.

Related Topics

- [Populating a SoundBank](#)
- [Deleting SoundBanks](#)
- [Monitoring the Details of a SoundBank](#)

Deleting SoundBanks

You can delete a SoundBank when you no longer need it.

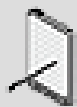
To delete a SoundBank:

1. In the **Project Explorer**, switch to the **SoundBanks** tab.
2. Do one of the following:

Right-click the SoundBank that you want to delete and select **Delete Selection**.

Click the SoundBank that you want to delete and press the **Delete** key.

The selected SoundBank is deleted.



Note

If you delete a SoundBank by mistake, you can undo the delete by pressing **Ctrl+Z**, or by clicking **Edit > Undo**.

Related Topics

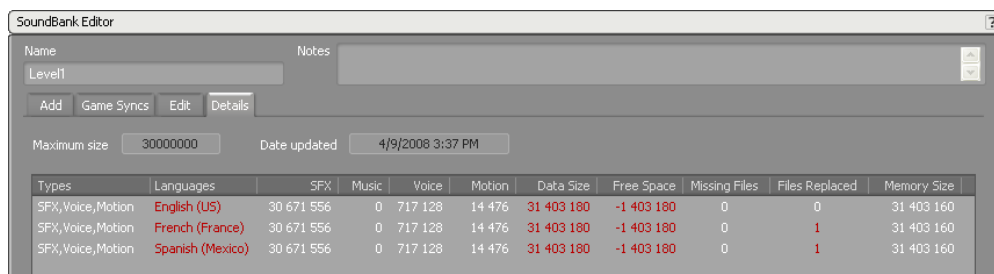
- [Populating a SoundBank](#)
- [Renaming a SoundBank](#)
- [Monitoring the Details of a SoundBank](#)

Monitoring the Details of a SoundBank

Since it is crucial to respect the memory limitations of each platform, it is important that you can manage and troubleshoot each of the SoundBanks

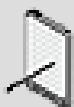
in your project. You need to determine which SoundBanks are over their maximum size, which language files are missing, the total size of the generated SoundBank, among other important information that is required to manage your SoundBanks effectively.

Wwise displays these details on the Details tab of the SoundBank Editor.



Types	Languages	SFX	Music	Voice	Motion	Data Size	Free Space	Missing Files	Files Replaced	Memory Size
SFX, Voice, Motion	English (US)	30 671 556	0	717 128	14 476	31 403 180	-1 403 180	0	0	31 403 160
SFX, Voice, Motion	French (France)	30 671 556	0	717 128	14 476	31 403 180	-1 403 180	0	1	31 403 160
SFX, Voice, Motion	Spanish (Mexico)	30 671 556	0	717 128	14 476	31 403 180	-1 403 180	0	1	31 403 160

If your SoundBank contains Sound Voice objects, the information on the Details tab is organized by language. Wwise also displays some information in red to help you quickly identify potential problems.



Note

All memory information in the SoundBank Manager and SoundBank Editor is displayed in bytes.

To monitor the details of a SoundBank:

1. In the SoundBank Manager, double-click the SoundBank whose details you want to review.

The related information is displayed on the Details tab of the SoundBank Editor.

2. Review the contents of the Details tab for information about the following:

Types - The types of objects included in the SoundBank, such as SFX, Voice, Music, and Motion.

Languages - The language of the corresponding SoundBank.

SFX - The amount of memory occupied by the Sound SFX objects.

Music - The amount of memory occupied by the music objects.

Voice - The amount of memory occupied by the Sound Voice objects for a particular language.

Motion - The amount of memory occupied by the Sound Voice objects for a particular language.

Data Size - The total amount of memory occupied by the SFX, Voice, and Motion objects for a particular language.

Free Space - The amount of space remaining in the SoundBank. This value is only displayed when a maximum size is specified.

Missing Files - The number of media files missing from a particular language.

Files Replaced - The number of missing Sound Voice audio files that are currently replaced by the audio files of the Reference Language.

Memory Size - The amount of space occupied by the SoundBank data that is to be loaded into memory.

Prefetch Size - The amount of space occupied by the prefetch data of the streamed media files. This information is loaded into memory with the SoundBank to ensure that the streamed media files playback without delays.

File Size - The total size of the generated SoundBank file.

Related Topics

- [Populating a SoundBank](#)
- [Managing the Contents of a SoundBank](#)
- [Deleting SoundBanks](#)

Defining Custom Attributes for Your SoundBanks

Before generating the SoundBanks for each of your platforms and languages, you need to define the settings for your SoundBanks. These settings are defined at the project level in the Project Settings dialog box, but you can also override these settings to create custom user settings. The following SoundBank user settings can be defined:

- [Defining Custom SoundBank Settings](#) - determine what information is to be included with the generated SoundBanks.
- [Specifying a Custom Location for Your Saved SoundBanks](#) - determine the location on your hard drive or network where the SoundBanks will be saved.
- [Defining Custom User Steps to be Performed Pre/Post SoundBank Generation](#) - determine which tasks will be performed immediately before the SoundBanks are generated.

- [Defining Custom User Steps to be Performed Pre/Post SoundBank Generation](#) - determine which tasks will be performed immediately after the SoundBanks are generated.
- [Specifying the Input/Output Location for External Source Files](#) - specify the input and output paths for the external audio sources that will be used with the Template Source plug-in.

For information on setting these attributes at the project level, refer to [Defining the SoundBank Settings for Your Project](#).

Defining Custom SoundBank Settings

The SoundBank settings determine what information will be part of the generation process, how it will be included, and in what format it will be generated. Although this information is defined at the project level, there may be circumstances where you will want to override it to create your own custom settings. The settings you choose will depend on how the data and media within the SoundBanks are accessed by your game.



Note

These custom settings will only be available to you and will only affect the SoundBanks that you generate.

To define custom SoundBank user settings:

1. In the SoundBank Manager, click **User Settings**.

The SoundBank User Settings dialog box opens.

2. Select the **Override Project SoundBank Settings** option.
3. Select any of the following options to define custom settings for your SoundBanks:

Allow SoundBanks to exceed maximum size to generate SoundBanks even if they exceed the maximum size specified.

Generate SoundBank content files to create files that list the contents of each SoundBank. The content files include information on events, busses, states, and switches, as well as a complete list of streamed and in memory media files.

Generate header file to create a header file that maps event, state, switch, and game parameter names to IDs.

Max attenuation to include maximum attenuation information in the SoundBanksInfo.xml file for each event.

Estimated duration to include the estimated maximum and minimum duration for each event, as well as whether a sound loops infinitely or is a one-shot sound, in the SoundBanksInfo.xml file for each event.

Use **SoundBank Names** to use SoundBank names (checked) or IDs (unchecked) to name generated .bnk SoundBank files and to reference one bank within another bank.

4. If you chose to generate a header file, you must decide where it will be saved. To do so, do the following:

Type a path directly in the text box.

Click the **Browse** button (...) and use the browser to navigate to the location of your choice.



Note

You can use a full path or a relative path to specify the location where the header file will be saved. When using a relative path, use the project folder as the origin of the path.

5. If you chose to generate SoundBank content files, you can select the desired text file format with the **SoundBank content file format** option.



Tip

If you have file paths, object names or object notes that contain non-ANSI characters, you should use the Unicode format.

6. Click OK to apply the settings.



Note

These custom user settings will be used to generate your SoundBanks until you remove the override.

Related Topics

- [Specifying a Custom Location for Your Saved SoundBanks](#)
- [Defining Custom User Steps to be Performed Pre/Post SoundBank Generation](#)

Specifying a Custom Location for Your Saved SoundBanks

All the SoundBanks that are generated for the project are automatically saved in the default project location, which is specified on the SoundBanks tab of the **Project Settings** dialog box. If you need to override this location, you can by creating a custom user setting.

When specifying the location for your saved SoundBanks, you can use a full path or a relative path. When using a relative path, use the project folder as the origin of the path. For example, the following full path and relative path specify the same location:

- C:\Wwise Projects\My Project\GeneratedSoundBanks\Windows
- GeneratedSoundBanks\Windows\

To specify a new location for your saved SoundBanks:

1. In the SoundBank Manager, click **User Settings**.

The SoundBank User Settings dialog box opens.

2. Select the **Override Project SoundBank Paths** option.
3. Specify a path by doing one of the following:

Type a path directly in the text box.

Click the **Browse** button (...) and use the browser to navigate to the location of your choice.

4. Click **OK** to apply any changes you made.



Note

Your generated SoundBanks will be saved in this custom location until you remove the override.

Related Topics

- [Defining Custom SoundBank Settings](#)
- [Defining Custom User Steps to be Performed Pre/Post SoundBank Generation](#)

Defining Custom User Steps to be Performed Pre/Post SoundBank Generation

Depending on your workflow, you may have a certain step or task that needs to be performed immediately before or immediately following the generation of your SoundBanks. For example, you may want to check out specific SoundBank files from your source control system before generating them or you may

want the streamed files to be copied to the SoundBanks directory immediately following generation. Although this information is usually defined at the project level, there may be circumstances where you will want to override it to create your own custom settings.

In Wwise, these types of tasks are defined by creating command lines. A special command line editor exists within Wwise making it easy for you to build as many command lines as you need. To simplify the process even further, the editor contains a list of all the Wwise-specific and other Windows environmental variables that can be used in a command line.





Note

The build process may be interrupted if an external tool returns an error or cannot be found. This is accomplished by setting the corresponding log severity level to **Fatal Error** in the log settings. For more information, refer to [Changing the Severity of Messages in the Log](#).

When a custom **Global opening step** fails, the entire build process is interrupted and no banks are created. On the other hand, when the failure occurs on a custom step for a specific platform, only that platform is skipped. However, the **Global closing step** will be skipped since part of the process wasn't successful. To be considered failing, an external process has to return a value other than zero.

The specific Wwise variables available for writing custom command lines are as follows:

Command Line Variable	Description
\$(AllowExceedMaximum)	Specifies whether SoundBanks can be generated even if they exceed the maximum size specified. This variable is set to true when the Allow SoundBanks to exceed maximum option is selected.
\$(ContentFileFormat)	Specifies the file type for generated SoundBank content files. Possible values are: <ul style="list-style-type: none"> • ANSI • Unicode
\$(GenerateContentFile)	Specifies whether files that list the contents of each SoundBank are created. The content files include information on events, busses, states, and switches, as well as a complete list of streamed and in memory audio files. This variable is set to true when the Generate SoundBank content files option is selected.

Command Line Variable	Description
\$(GenerateHeaderFile)	<p>Specifies whether a header file is generated that maps event, state, switch, and game parameter names to IDs.</p> <p>This variable is set to true when the Generate header file option is selected.</p>
\$(GenerateMaxAttenuationInfo)	<p>Specifies whether the maximum attenuation information is generated for events.</p> <p>This variable is set to true when the Event Info: Max attenuation option is selected.</p>
\$(GenerateEstimatedDuration)	<p>Specifies whether the estimated maximum and minimum duration and duration type information is generated for events.</p> <p>This variable is set to true when the Event Info: Estimated Duration option is selected.</p>
\$(HeaderFileFullPath)	The full path of the header file, which is: \$(HeaderFilePath)\Wwise_IDs.h.
\$(HeaderFilePath)	<p>The path or location where the header file will be saved.</p> <p>This path is taken from the Header file path text box.</p>
\$(InfoFilePath)	The full file name of the current platform's Info file.
\$(IsRunningFromCmdLine)	Specifies whether Wwise was launched from the command line with the “-generatesoundbanks” flag.
\$(LanguageList)	<p>The list of languages passed to the command line OR the selected languages in the SoundBank Manager.</p> <div style="background-color: #f0f0f0; padding: 10px; border: 1px solid #ccc;">  <p>Note</p> <p>The list is a space-separated list.</p> </div>
\$(Platform)	The name of the current platform.
\$(SoundBankList)	<p>The list of SoundBanks passed to the command line OR the selected SoundBanks in the SoundBank Manager.</p> <div style="background-color: #f0f0f0; padding: 10px; border: 1px solid #ccc;">  <p>Note</p> <p>The list is a space-separated list. Use double-quotes to enclose the list in one single argument.</p> </div>
\$(SoundBankPath)	The path or location where the current platform's SoundBanks are saved.
\$(UseSoundBankNames)	<p>Specifies whether SoundBank names (true) or IDs (false) are used to name generated SoundBank BNK files, as well as within banks to refer to media in other banks.</p> <p>This variable is set to true when the Use SoundBank names option is selected.</p>
\$(WwiseExeDriveLetter)	The drive letter on your workstation where the Wwise executable (Wwise.exe) is located.
\$(WwiseExePath)	The path or location of the Wwise executable (Wwise.exe).

Command Line Variable	Description
\$(WwiseExeProcessID)	The numerical Process Identifier of the Wwise executable (Wwise.exe).
\$(WwiseProjectDriveLetter)	The drive letter on your workstation where the Wwise project is located.
\$(WwiseProjectName)	The name of the current project.
\$(WwiseProjectPath)	The path or location of the Wwise project.



Note

Environment variables are automatically mapped, for example, \$(WWISESDK).

To be as flexible as possible, Wwise allows to define different command lines for the following types of steps:

- **Global opening step** - A command line that applies to all platforms and is performed before any other step.
- **Platform-specific pre-generation step** - A command line that applies to a specific platform and is performed before the SoundBanks are generated.
- **Platform-specific post-generation step** - A command line that applies to a specific platform and is performed after the SoundBanks are generated.
- **Global closing step** - A command line that applies to all platforms and is performed after all other steps.

By default, every project includes a post-generation step command line that uses the CopyStreamedFiles tool to copy the streamed files in your project to the SoundBank directory. For more information about this external tool, refer to [Using the CopyStreamedFiles Tool](#). You can, however, automate any type of task by executing a different command line. Wwise also ships with another Factory command line that uses the File Packager to generate a package containing all data and media within your SoundBanks. For more information about the File Packager, refer to [Chapter 35, Managing File Packages](#). For more information about loading factory command lines, refer to [Loading Factory/Custom Command Lines](#).

You can also save the command lines you create to a file (.wcmdline) so that you can use them later on, within the same project, across projects, or if you want to share them with other users. For more information on saving commands, refer to [Saving Custom Command Lines to a File](#).

To define user tasks to be performed pre SoundBank generation:

1. In the SoundBank Manager, click **User Settings**.

The SoundBank User Settings dialog box opens.

2. Select the **Override Project Pre-Generation Step** option.
3. To add to or modify the Global opening pre-generation step that is defined in the Project Settings dialog box, click the corresponding **Edit** button (...).

The Pre-Generation Step Editor opens.

4. In the Description text box, type a name that clearly describes the step or task that will be performed.
5. In the Commands text box, write a new command line or edit the existing command line, as required.



Note

The Commands text box works like most other text editors, which means you can add new lines of text by pressing Enter, delete text by selecting it and pressing Delete, and so on.

6. If you want to insert built-in macros and environment variables in your command, do the following:

In the Macros group box, select one of the following options:

Built-in Macros - To display a list of Wwise-specific variables that can be used within the Wwise command lines.

Environment Variables - To display a list of Windows-specific environment variables that can be used within the Wwise command lines.

To add a variable to the command line, do one of the following:

Double-click a variable in list.

Select a variable from the list and then click **Insert**.

Continue to add variables to your command line, as required.

7. If you want to add another pre-generation step, simply go to the end of the first line, press **Enter**, and then start creating a new command line.
8. Click **OK** to save the command line and to close the Pre-Generation Step Editor.



Note

If you want to save the command line to file, click the Save As button in the Editor. For more information on saving custom

command lines, refer to [Saving Custom Command Lines to a File](#).

9. To add or modify the existing platform-specific pre-generation steps, repeat steps 3-8 for each platform.



Note

You can also load factory and previously saved custom command lines into the Editor by clicking the Load button. For more information on loading factory/custom commands, refer to [Loading Factory/Custom Command Lines](#).

To define user tasks to be performed post SoundBank generation:

1. In the SoundBank Manager, click **User Settings**.

The SoundBank User Settings dialog box opens.

2. Select the **Override Project Post-Generation Step** option.
3. To add or modify the existing post-generation step that is defined in the Project Settings dialog box, click one of the **Edit** buttons (...).

The Post-Generation Step Editor opens.

4. In the Description text box, type a name that clearly describes the step(s) or task(s) that will be performed.
5. In the Commands text box, write a new command line or edit the current command line, as required.



Note

The Commands text box works like most other text editors, which means you can add new lines of text by pressing Enter, delete text by selecting it and pressing Delete, and so on.

6. If you want to insert built-in macros and environment variables in your command, do the following:

In the Macros group box, select one of the following options:

Built-in Macros - To display a list of Wwise-specific variables that can be used within the Wwise command lines.

Environment Variables - To display a list of Windows-specific environment variables that can be used within the Wwise command lines.

To add a variable to the command line, do one of the following:

Double-click a variable in list.

Select a variable from the list and then click **Insert**.

Continue to add variables to your command line, as required.

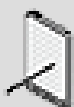
7. If you want to add another pre-generation step, simply go to the end of the first line, press **Enter**, and then start creating a new command line.
8. Click **OK** to save the command line and to close the Post-Generation Step Editor.



Note

If you want to save the command line to file, click the Save As button in the Editor. For more information on saving custom command lines, refer to [Saving Custom Command Lines to a File](#).

9. Repeat steps 3-8 for the global closing step and/or for each additional platform.



Note

You can also load factory and previously saved custom command lines into the Editor by clicking the Load button. For more information on loading factory/custom commands, refer to [Loading Factory/Custom Command Lines](#).

Related Topics

- [Defining Custom SoundBank Settings](#)
- [Specifying a Custom Location for Your Saved SoundBanks](#)
- [Loading Factory/Custom Command Lines](#)
- [Saving Custom Command Lines to a File](#)

Specifying the Input/Output Location for External Source Files

If you plan to use the Template Source plug-in you must specify the location of the external audio files that will be associated with the template source at runtime. You must also specify the folder in which the converted sources will be saved so that they can be used by Wwise while the game is being played.

To specify the input/output paths for external audio sources:

1. In the SoundBank Manager, click **User Settings**.

The SoundBank Settings dialog box opens.

2. Switch to the **External Sources** tab.
3. If you want to convert the external sources as part of the SoundBank generation process, select the **Convert external sources when generating banks** option.
4. To modify the input path specified in the Project Settings dialog box, select the **Override Input Path** option.
5. Specify a folder where the external source files are located by doing one of the following:

Click in the External Sources list and type the path where the audio files are located.

Click the **Browse** button (...) that corresponds to one of the game platforms, navigate to the folder that contains the audio files for that platform, and then click **Open**.

6. Repeat step 3 for each of the platforms.
7. To modify the output path specified in the Project Settings dialog box, select the **Override Output Path** option.
8. Specify a folder where the external sources will be saved by doing one of the following:

Click in the External Sources Output Folder list and type the path where you want the audio files to be saved.

Click the **Browse** button (...) that corresponds to one of the game platforms, navigate to the folder where you want to save the audio files for that platform, and then click **OK**.

9. Repeat step 5 for each of the platforms.
10. Click **OK** to close the SoundBank Settings dialog box.

Generating SoundBanks for a Project

You can generate SoundBanks for each game platform and language that you intend to create. SoundBanks can be generated at any point in the development process so that you can test how they integrate into the various game platforms. For added convenience, Wwise also allows you to generate the SoundBanks for all platforms and all languages simultaneously.

When a SoundBank is generated, it can include any of the following information:

- Event information
- Sound, music, motion, container, and actor-mixer information
- Sound, music, or motion data for in-memory media
- Prefetch data for streamed media files with zero-latency
- References to streamed media files

The information contained in the SoundBanks is project exclusive, which means that a SoundBank can only be used with other SoundBanks generated from the same project.

If your SoundBanks contain events or object structures that have been deleted from your project, you will still be able to generate your SoundBanks. These invalid project elements are ignored by Wwise during the generation process and will not cause errors or take up additional space. You should, however, remove these invalid events and object structures from your SoundBanks in order to maintain the integrity of your project.

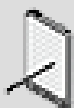


Note

When an event or object structure within a SoundBank becomes invalid, the corresponding icon is removed and the word “Missing” appears after its name on the Add tab of the SoundBank Editor.

Before generating your SoundBanks, you need to set a series of options that define whether content and header files will be generated, whether SoundBank names can be used, the location where your SoundBanks will be saved, whether streamed files need to be copied to the SoundBank directory, among other things. These SoundBank settings are defined at two separate levels within Wwise:

- **Project level** - To create default settings for your project. These settings are defined in the Project Settings dialog box. For more information on defining the SoundBank settings at the project level, refer to [Defining the SoundBank Settings for Your Project](#).
- **User level** - To create custom user settings that override the project settings. User settings are defined in the SoundBank User Settings dialog box. For more information on defining custom user settings, refer to [Defining Custom SoundBank Settings](#).



Note

Before you can generate SoundBanks for a particular console, such as the Xbox 360 or PlayStation 3, you need to be an

authorized content creator/developer for that platform and have the necessary platform .cab file. If your installation package did not include the .cab file for a particular platform, the platform will not appear as one of the available options in the SoundBank Platforms list. For detailed information about how to install Wwise and the associated .cab files, refer to the Installation Issues section of the Wwise Installation and Migration Guide.

To generate the SoundBanks for your project:

1. Switch to the SoundBank layout by doing one of the following:

From the menu bar, click **Layouts > SoundBank**.

Press F7.



Tip

You can open a floating view of the SoundBank Manager from any other layout by clicking **Views > SoundBank Manager** or by using the shortcut key (Shift+B).

2. In the SoundBank Manager, select the SoundBanks that you want to generate.
3. From the Platforms list, select the platforms for which you want to generate SoundBanks.
4. From the Languages list, select the languages for which you want to generate SoundBanks.
5. To begin generating the SoundBanks, click **Generate**.

The Generating SoundBanks dialog opens where you can view the progress of the SoundBank generation process. When the SoundBanks are generated, the SoundBanks Generation - Completed dialog box opens.



Note

SoundBanks that were previously generated and have not changed will not be regenerated. In this case, "Up to date" appears in the **Created** column of the **Results** panel.

6. Review the messages in the dialog box to ensure that all SoundBanks have been generated successfully.
7. As a final verification, review the entries in the log. The SoundBank log contains a complete list of errors, warnings, and other messages that were

encountered during the generation process. For more information on the SoundBank Log, refer to [Troubleshooting SoundBank Issues Encountered During Generation](#).

8. When you are finished reviewing the SoundBank Log, you can close it along with the Generating SoundBanks - Completed dialog box.

The SoundBank files are located in the folder that you specified, and can now be integrated into your game.

Related Topics

- [Defining the SoundBank Settings for Your Project](#)
- [Defining Custom Attributes for Your SoundBanks](#)
- [Renaming a SoundBank](#)
- [Monitoring the Details of a SoundBank](#)
- [Deleting SoundBanks](#)
- [Using Scripts to Generate SoundBanks](#)
- [Searching for Elements within a SoundBank](#)
- [Troubleshooting SoundBank Issues Encountered During Generation](#)

Troubleshooting SoundBank Issues Encountered During Generation

Adding Messages to the Log Ignore List

There may be situations where you no longer want or need specific warnings and/or messages to be displayed in the SoundBank Log during the generation process. To prevent these types of warnings and messages from appearing in the SoundBank Log, you can add them to the Log Ignore list. When you want to include these messages back in the log, you can simply remove the message types from the Log Ignore List.

You can also define which messages will appear in the Log Ignore List in the Project Settings dialog box. For more information, refer to [Managing Messages that Appear in the Logs](#).

To add messages to the Log Ignore list:

1. In the SoundBank Log, select the message types that you no longer want to appear in the log.
2. Right-click the selection and select **Add Message(s) to Ignore List**.

The next time you generate the SoundBanks for your project, these types of messages will no longer appear in the SoundBank Log.

Related Topics

- [Troubleshooting SoundBank Issues Encountered During Generation](#)
- [Generating SoundBanks for a Project](#)
- [Using Scripts to Generate SoundBanks](#)

If Wwise encounters any issues while generating your SoundBanks, it displays information about each issue in the SoundBank Log. While you are generating your SoundBanks, you can view the log as it appears in the bottom panel of the Generating SoundBanks dialog box. You can also review the information in the log at a later time using the SoundBank Log view.

In certain situations, you may want to prevent certain messages from appearing in the SoundBank log. You can do this by adding the messages you don't want to appear to the Log Ignore List. For more information on the Log Ignore List, refer to [Adding Messages to the Log Ignore List](#). You can also limit the number of times a message will be displayed in the Log. For information on setting this limit, refer to [Managing Messages that Appear in the Logs](#).

Related Topics

- [Adding Messages to the Log Ignore List](#)
- [Generating SoundBanks for a Project](#)
- [Using Scripts to Generate SoundBanks](#)

Changing the Severity of Messages in the Log

There may be situations where you want to change the severity of a specific message, warning or error displayed in the SoundBank Log. You can change the severity of a message in the Project Settings dialog box, Log tab.

For more information, refer to [Managing Messages that Appear in the Logs](#).

Related Topics

- [Troubleshooting SoundBank Issues Encountered During Generation](#)
- [Generating SoundBanks for a Project](#)
- [Using Scripts to Generate SoundBanks](#)



Using Scripts to Generate SoundBanks

If you plan to generate your SoundBanks on a regular basis, you may want to create a script to automate the generation process. To accomplish this, you can use the Wwise SDK to generate SoundBanks from the command line. For a full description of this functionality, refer to the section [Wwise SDK > Going Further > Generating Banks from the Command Line](#) in the Wwise SDK documentation.

Using the CopyStreamedFiles Tool

The CopyStreamedFiles tool is a stand-alone utility that copies the streamed media files in your project to the location where your SoundBanks are saved (by default, SoundBanks are saved in the platform-specific folder in the GeneratedSoundBanks directory). When this tool is called from the command line, it scans the SoundBanksInfo.xml file to find out which files are streamed and then copies them from your project cache folder to the appropriate folder in the SoundBank directory. By default, every project includes a post-generation step command line that calls the CopyStreamedFiles tool to complete this task, but you can modify this step or create new steps to meet the particular needs of your project.

The CopyStreamedFiles tool is located in the Wwise executable directory. When calling this tool from the command line, you need to define certain parameters, including the location of the SoundBanksInfo.xml file, and the location where the streamed files will be copied. The following table lists the parameters that can be used within the command line.

Parameter	Description
-info <file path>	Specifies the location of the SoundBanksInfo.xml file. This option is set to “\$(InfoFilePath)” (with quotes) when used in Wwise as a post-generation step.
-outputpath <folder path>	Specifies the base folder where streamed files will be copied, which should be the base folder where SoundBanks are generated. Set to “\$(SoundBankPath)” (with quotes) when used in Wwise as a post-generation step.
-languages “language1 languageN“	<p>Specifies the list of languages to copy.</p> <div style="background-color: #f0f0f0; padding: 10px; border: 1px solid #ccc;">  <p>Note</p> <p>The list is space-separated. Use double-quotes to enclose the list in one single argument. Use the keyword “SFX“ in the language list to copy them as well.</p> </div> <p>This argument is optional. By default all languages are copied.</p>
-banks “soundbank1 soundbankN“	<p>Specifies the list of SoundBanks for which to copy streamed files. The names do not include the bnk extension. Alternatively, a text file containing a list of SoundBank names can be specified, by passing in the full path to the text file, including the extension “.txt“.</p> <div style="background-color: #f0f0f0; padding: 10px; border: 1px solid #ccc;">  <p>Note</p> <p>The list is space-separated. Use double-quotes to enclose the list in one single argument.</p> </div>

Parameter	Description
	This argument is optional. By default all streamed files are copied.
-verbose	Enable extra console text output. This argument is optional.

Related Topics

- [Defining Custom Attributes for Your SoundBanks](#)
- [Generating SoundBanks for a Project](#)
- [Chapter 35, *Managing File Packages*](#)

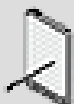
Strategies for Managing SoundBanks

The following sections describe the different methods that can be used to generate and integrate the banks in your game. In a single game, you can use one or a combination of the different methods. Since every game is different, the method or methods you choose will depend on the specific requirements of your game.

The choices you make when creating your banks will have a significant impact on the amount of work that will be required to manage the audio and motion assets in the game and will have a direct impact on the performance of your game. It is highly recommended that both the sound designer and programmer review this section carefully so that both are aware of the possibilities that are available. Working together, you can come up with a strategy that meets the specific needs of your game. Remember that all solutions will work, but the strategy you choose should take into consideration the memory usage, the I/O access, and the ease of integration in game. Each method has its advantages and drawbacks, so in most situations, it will be a question of balance between memory usage and ease of integration.

The following methods are described in the following sections:

- [Method 1: The “All-in-one” Bank](#)
- [Method 2: Multiple Complete Banks](#)
- [Method 3: Micromanaging Your Media](#)
- [Method 4: Preparing Action Events](#)
- [Method 5: Preparing Events and Game Syncs \(Switches and States\)](#)



Note

Two other methods also exist, which combine the benefits of the existing LoadBank and PrepareEvent methods. Using these two methods, entire banks are prepared, which means you can combine all data and media within the same bank, avoid media duplication in memory, and still load media only when required.

For more information on preparing banks, refer to the section “Sound Engine Integration Walkthrough > Integrate Wwise Elements Into Your Game > Integrating Banks > Integration Details - Banks > Loading Banks > Preparing Banks” in the [Wwise SDK documentation](#).

Method 1: The “All-in-one” Bank

This method will apply if:

- The game has a limited amount of audio and/or motion assets.
- The game has plenty of free memory.

Of course, most games generally don't have memory to waste, but this technique has the major advantage of being very simple to use and to maintain. The main reason for using this technique is to have the entire Wwise project integrated in the game in a minimum amount of time.

To create the “All-in-one” bank in Wwise:

1. Create one SoundBank, and name it appropriately.
2. Load the bank into the SoundBank Editor and then drag the following items to the “Add” tab:

The “Actor-Mixer Hierarchy\Default Work Unit”

The “\Events\Default Work Unit”

The “\Interactive Music Hierarchy\Default Work Unit” (Only required if using interactive music)

The “\Dynamic Dialogue\Default Work Unit” (Only required if using dynamic dialogue)

The check boxes under the Events, Structures, and Media columns are all selected by default. This is good because, for this method, we want everything to be included in the bank.



Note

In a new project, only the default work unit is available. If more work units are created, these work units should also be added to the banks as required.

3. After all the work units have been added to the SoundBank, you can generate it and then copy the generated bank folder to the game application.

To integrate the “All-in-one” bank in game:

1. Since there is only one SoundBank for this game, you can simply load it when initializing the game. Of course, the sound engine must be correctly initialized first.

The following sample of code gives you an idea how the “All-in-one” bank can be loaded in game:

```

...
// Initialize the sound engine here.
...

// Load the Init bank and the "All in one" bank.
AkBankID bankID; // not used in this sample.
AKRESULT eResult = AK::SoundEngine::LoadBank( L"Init.bnk", AK_DEFAULT_POOL_ID, bankID );
if( eResult == AK_Success )
{
    eResult = AK::SoundEngine::LoadBank( L"MyAllInOneBank.bnk", AK_DEFAULT_POOL_ID, bankID );
}
...

```

Additional Notes on this Method

The following table lists the pros and cons of the “All-in-one” bank method.

Pros	Cons
Easiest way to maintain the content of the banks from the sound designer point of view.	Inefficient usage of memory because all events, structures, and in-memory media for the entire game are loaded at all times.
No need to recompile the game when changing the content of the banks since the game always loads the same bank.	
No complex in-game loading and unloading of banks.	
No need to track in-game which sounds are available or not.	

Even though this method is a good way to quickly and easily integrate the audio and motion in your game, don't wait until the end of the project to switch to a method that makes better usage of your game's memory.

Method 2: Multiple Complete Banks

This method will apply if:

- The game or its audio/motion content can be split into multiple sections.

This method works well for single player games, where all possible sounds and motion are only driven by the current location of the player in the game. By splitting the content into multiple banks, you can manage memory more efficiently than the first method, yet still benefit from a relatively easy integration of audio and motion in your game.

To begin with, you must first determine how to split up your banks. For example, you may decide to split up your banks as follows:

- One general bank containing all the events that can occur at any point in the game. This bank would be loaded into memory at all times.
- One bank per level, or one per environment. For sounds and motion that depend on the actual location of the main character.
- And possibly some additional banks, depending on the requirements of your particular game.

To create multiple complete banks in Wwise:

1. Create the banks you will need for the game and then name them appropriately, for example, “CommonEvents”, “Level_1”, “Level_2”, and “Level_3”.
2. Group your events into virtual folders in Wwise. Create one virtual folder per bank and then drag each virtual folder into its corresponding bank. By adding virtual folders to your SoundBanks, you can avoid having to edit the contents of your SoundBanks every time a new event is added to your project. When the contents of your virtual folder change, the SoundBank automatically gets updated.
3. Add all events to their respective banks. This step is only required to add events that fall outside the original folders, if any. If an event needs to be in multiple banks, then simply add it to the required banks.
4. Generate the banks and copy the generated bank folder to the game application.

To integrate the multiple complete banks in game:

1. In the game, simply load the right bank at the right time. For example, the game could load the general bank at the beginning and then load the other banks based on the player's actual location in the game. Note that some games will need to have enough memory to load more than one “level” at a time, to allow for transitions between levels.

The following sample of code gives you an idea how these banks can be loaded in game.

```

...
// Initialize the sound engine here.
...

// Load Init bank and Common bank.
AkBankID bankID; // Not used in this sample.
AKRESULT eResult = AK::SoundEngine::LoadBank( L"Init.bnk", AK_DEFAULT_POOL_ID, bankID );
if( eResult == AK_Success )
{
    eResult = AK::SoundEngine::LoadBank( L"CommonEvents.bnk", AK_DEFAULT_POOL_ID, bankID );
}

...
// And at various places in the code, based on the actual needs:
eResult = AK::SoundEngine::LoadBank( L"Level_1.bnk", AK_DEFAULT_POOL_ID, bankID );
...
eResult = AK::SoundEngine::LoadBank( L"Level_2.bnk", AK_DEFAULT_POOL_ID, bankID );
...
eResult = AK::SoundEngine::LoadBank( L"Level_3.bnk", AK_DEFAULT_POOL_ID, bankID );
...
eResult = AK::SoundEngine::UnloadBank( L"Level_1.bnk" );
...
eResult = AK::SoundEngine::UnloadBank( L"Level_2.bnk" );
...
eResult = AK::SoundEngine::UnloadBank( L"Level_3.bnk" );

```



Note

As an alternative to the LoadBank command, you could prepare your banks using the AkBankContent_All command. The advantage of preparing your banks is that you avoid the possibility of duplicating media in memory. Refer to the section “Preparing Banks” in the [Wwise SDK documentation](#).

Additional Notes on this Method

The following table lists the pros and cons of the “All-in-one” bank method.

Pros	Cons
<p>May require a lot less memory than the “All-in-one” bank technique.</p> <p>Very easy to integrate in game.</p>	<p>Not well adapted for online or event-based games, where the audio or motion requirements are driven by more than just simple facts like the main character's location.</p> <p>May cause some media file duplication loaded in memory, because banks can contain duplicated data.</p> <p>May increase the total space of the banks on disk because different banks may have similar content.</p>

Method 3: Micromanaging Your Media

This method will apply if:

- The game has a large number of in-memory media assets.
- The media requirements are difficult to predict in advance by the sound designer.

- The project uses switches and states to determine which sound or motion FX will play for a specific event.
- Sounds and motion FX can't easily be split into defined sections.

Games can be very complex and the triggering of sounds and motion can be based on a variety of different factors, including game textures, the time of day, the movement of game objects, and in some cases the actions of other players for online multiplayer games. In an event-based or object-based environment, sounds and motion FX could, for example, be loaded into memory based on the proximity of some other game objects. Every game object could have a list of banks that must be loaded when they are within a given range or simply because they exist.

Also, switches and states can determine which objects are played. When an event playing such objects is added to a bank, all possible media is automatically added to the bank as well. For example, you may have a single event called “Play_Footstep”. This event will play the appropriate sound based on the current texture of the ground, which is specified by changing the switch. Although this works well, it can be a waste of memory keeping the sounds “footstep_sand.wav” or “footstep_winter.wav” ready in memory when the gameplay is happening inside a building in London.

To avoid wasting memory in this case, you can add events and/or object structures to a bank and then specify which corresponding objects will be included in the bank. For example, let's take footstep sounds on different textures. In this case, we could create the following banks:

- The “EventBank”, which would contain the “Play_Footstep” event and structure.
- The “Winter_Footstep_bank”, which would contain the media for the footsteps that only occur in some part of the game (winter).
- The “Desert_Footstep_bank”, which would contain the media for the footsteps that only occur in some part of the game (desert).
- The “Common_Footstep_bank”, which would contain the media for the footsteps that can occur anywhere (wood, concrete, and so on).

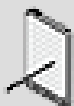
Creating Banks in Wwise that Micromanage your Media

Let's say you have three different textures in your game (snow, sand, and concrete). In Wwise, you have a switch container, that plays one of three random containers, based on the switch “ground_texture”. Each of the three random containers has four variations of the footstep sound on a given texture.

To create banks in Wwise that micromanages your media:

1. Create a bank called “EventBank” and load it into the SoundBank Editor.
2. Drag the “Play_Footstep” event to the Add tab of the SoundBank Editor.

3. Deselect the “Media” check box for this event, but leave the Events and Structures check boxes selected.
4. Create the three other banks, one for each texture.
5. In each bank, drag the random container associated with this texture.



Note

It would work as well if instead of drag and dropping the random containers we would drag and drop each of the sounds individually. But the advantage of using the containers is that all the sounds in the container will automatically be added in the bank instead of having to make all the changes manually if the container's content changes.

6. Deselect the “Events” and “Structures” check boxes for each of the three texture banks, leaving only the Media type checked.
7. Generate the banks and copy the generated bank folder to the game application.

At this point, we have four banks, one that contains the event and the structure data related to the audio that is to be played, and three others that contain only media associated with a particular ground texture.

To integrate banks in game using the micromanaging strategy:

1. In the game, load the common banks at the beginning of your game and then simply load the other banks when they are required. For example, the game could load the event bank and the common footsteps bank at the beginning and then load the other banks based on the player's actual location in the game.

The following sample of code gives you an idea how these banks can be loaded in game.

```
// Load Init and the Eventbank
AkBankID bankID; // Not used in this sample.
AKRESULT eResult = AK::SoundEngine::LoadBank( L"Init.bnk", AK_DEFAULT_POOL_ID, bankID );
if( eResult == AK_Success )
{
    eResult = AK::SoundEngine::LoadBank( L"EventBank.bnk", AK_DEFAULT_POOL_ID, bankID );
}
if( eResult == AK_Success )
{
    eResult = AK::SoundEngine::LoadBank( L"Common_Footstep_bank.bnk", AK_DEFAULT_POOL_ID, bankID );
}

...
// And at various places in the code, possibly based on the location:
eResult = AK::SoundEngine::LoadBank( L"Winter_Footstep_bank.bnk", AK_DEFAULT_POOL_ID, bankID );
...
eResult = AK::SoundEngine::LoadBank( L"Desert_Footstep_bank.bnk", AK_DEFAULT_POOL_ID, bankID );
...
eResult = AK::SoundEngine::UnoadBank( L"Winter_Footstep_bank.bnk" );
...
eResult = AK::SoundEngine::UnoadBank( L"Desert_Footstep_bank.bnk" );
```

Additional Notes on this Method

This was only a very specific example of one of many possible things that can be done using this technique. Since it is possible to decide object by object and event by event what will be included in every bank, you have complete control over the contents of each bank. Although you could create a separate bank for each object in your game, this would be very difficult to maintain as every new sound or motion FX would require new code to load the bank at the right place in the game. The goal for each game is to try and find a good balance between granularity and ease of integration in the game.



Note

If you are interested in a sound by sound loading scheme, you may want to prepare events instead of loading entire banks. For more information on preparing events, refer to [Method 4: Preparing Action Events](#).

The following table lists the pros and cons of the “All-in-one” bank method.

Pros	Cons
<p>The best option to optimize memory usage.</p> <p>Gives you full control over what media is loaded at any point in the game.</p>	<p>Requires a lot of communication between the sound designer and the game developers to determine which bank must be loaded, and when.</p>

Method 4: Preparing Action Events

This method will apply if:

- A very small level of granularity is required for media to keep the memory usage low.
- You do not want to worry about managing which media assets must be assigned which bank.

What exactly is a prepared action event? When calling the PrepareEvent function, the system analyses the action event and makes sure that all the structure and media related to this event are loaded into memory. If they are not, the system will automatically stream from disk the missing information. An event is prepared until it is explicitly unprepared.



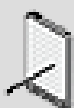
Note

Only action events can be prepared in advance. The “PrepareEvent” method does not work with dialogue events.

This method requires that you create at least one bank, however the structure part can be in either the same bank as the events, or in a completely separate bank.

When building banks that will use the PrepareEvent mechanism, the criteria is that every required event and structure must be found in at least one bank, and the loose media assets must be accessible by the file system. Remember though, that it is possible to manage your memory more efficiently by splitting structure into multiple banks.

Prior to preparing an action event, the event itself must have been loaded into memory from one bank (using LoadBank()). This is required because the event contains information about the dependencies required to prepare the event.



Note

It is also possible to prepare events in combination with `AK::SoundEngine::PrepareBank`. The main advantage of using the PrepareBank mechanism is that it removes the need to split the banks into "Event banks" and "Media banks". Under this method, all content is contained in the same bank, but when `AK::SoundEngine::PrepareBank` is called, only the metadata content of the bank is loaded into memory. When the media is required by your game, you can load it using PrepareEvent.

To setup your banks in Wwise when preparing events:

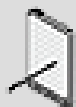
1. Create a bank called "Events" and load it into the SoundBank Editor.
2. Add some of the action events in your project to the "Event" bank, or simply add the event work units.
3. Deselect the "Media" check box, leaving only the "Events" and "Structures" check box selected. When using prepare event, the media must not reside in a bank, but be loaded directly off disk.



Note

For the purposes of this example, all the events and structures are contained in one bank. Although this is acceptable for small projects, you will most likely want to split up your content into several banks. It is also possible to create a separate "Structures" bank that does not need to be explicitly loaded nor prepared from the SDK command because each event includes references to the other banks that need to be loaded.

4. Generate the banks and copy the generated bank folder to the game application.



Note

The structure data contained within a single bank can't be split at run time. Therefore, if you are using `AK::SoundEngine::PrepareEvent`, and the structure data from a separate bank is required, all the structures within that bank will be loaded at once. For this reason, you may want to split the structure content in your project into multiple banks, to minimize the amount of unnecessary information that is loaded into memory.

To prepare events in game:

1. In the game, load the events bank at the beginning of your game and then prepare the events when they are required in game. The corresponding structures and media will be loaded automatically.

The following sample of code gives you an idea how to prepare events in game.

```

// Initializing the sound engine.

AkInitSettings initSettings;
AkPlatformInitSettings platformInitSettings;
AK::SoundEngine::GetDefaultInitSettings( initSettings );
AK::SoundEngine::GetDefaultPlatformInitSettings( platformInitSettings );

// Set the required settings

...

// Set PrepareEvent related settings
initSettings.bEnableGameSyncPreparation = false; // Not used in the current sample.

// (Optional) Allocate a memory pool into which prepared media will be loaded.
// If this is not done, memory will be allocated directly in the default memory pool.
{
    initSettings.uPrepareEventMemoryPoolID = AK::MemoryMgr.CreatePool( NULL, 4*1024*1024, 1024, AkMalloc );

    // (Optional) Give the memory pool a name. This can be very useful for profiling.
    AK::MemoryMgr::SetPoolName( initSettings.uPrepareEventMemoryPoolID, L"PrepareEventPool" );
}

AKRESULT eResult = AK::SoundEngine.Init( initSettings, platformInitSettings );
if( eResult != AK_Success )
{
    // Handle error.
}

// Load Init bank and the event/structure bank.
AkBankID bankID; // Not used in this sample.
AKRESULT eResult = AK::SoundEngine::LoadBank( L"Init.bnk", AK_DEFAULT_POOL_ID, bankID );
if( eResult == AK_Success )
{
    eResult = AK::SoundEngine::LoadBank( L"Events.bnk", AK_DEFAULT_POOL_ID, bankID );
}

...

// And then, at various points in the code:

AkIpCtstr pEventsNameArray[1] = { L"My_Event_Name" };

// Preparing an event:
eResult = AK::SoundEngine::PrepareEvent( Preparation_Load, pEventsNameArray, 1 ); // 1 is the array size

// Unpreparing an event:
eResult = AK::SoundEngine::PrepareEvent( Preparation_Unload, pEventsNameArray, 1 ); // 1 is the array size

```

Additional Notes on this Method

Keep in mind that calls to `AK::SoundEngine::PrepareEvent` must be considered as I/O functions calls. In the previous example, we used blocking functions. You can use other overloads of the function `AK::SoundEngine::PrepareEvent` to make them non blocking calls, and then revive the completion notification through a separate callback.

The following table lists the pros and cons of the “All-in-one” bank method.

Pros	Cons
Bank generation process is simple.	Potentially increases the number of reads and seeks on the disk as the media assets will be loaded one by one.
Level of granularity for media is very small.	
Maintain a low overall memory usage.	Less control over the total amount of memory used.
Easy to automate the process.	Not ideal for interactive music.

Method 5: Preparing Events and Game Syncs (Switches and States)

This method will apply if:

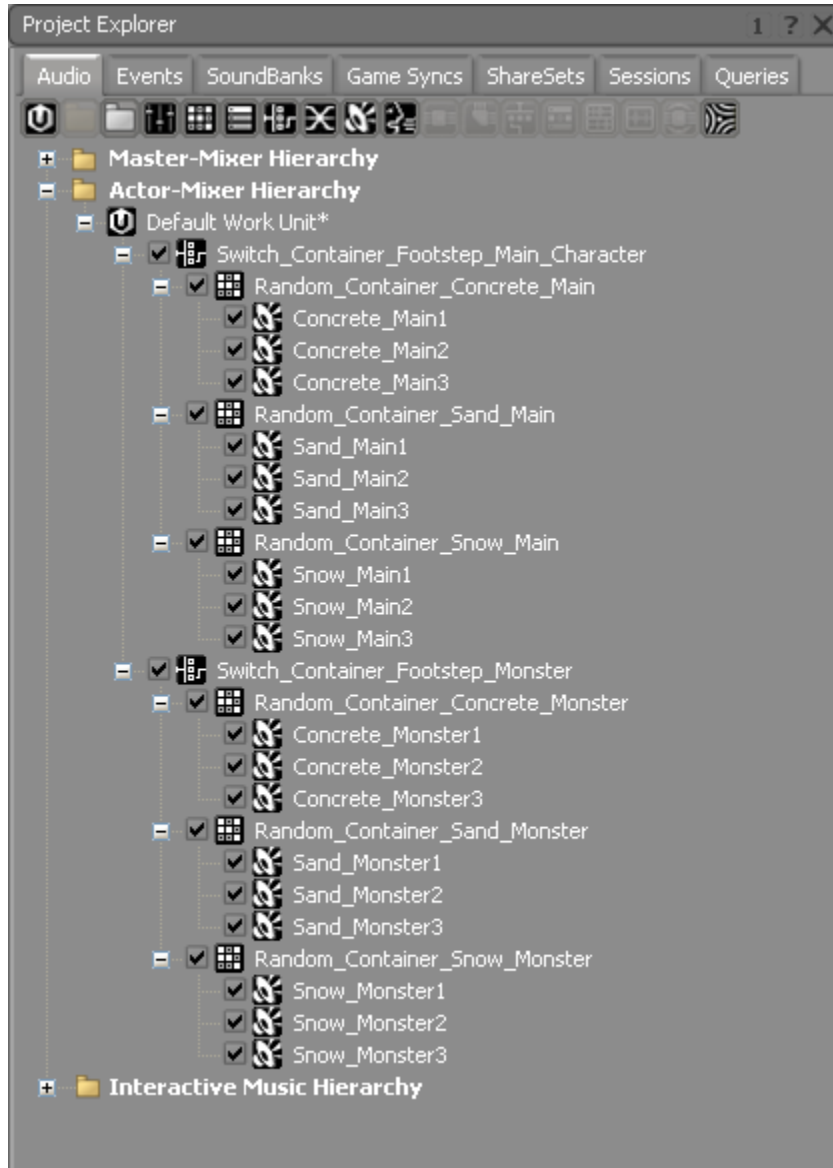
- A small level of granularity is required for media to keep the memory usage low.
- You do not want to worry about splitting your media assets into banks.
- You have events in your project that play different sounds and motion FX based on switches or states.
- Your project contains interactive music that plays music based on switches or states.

This method is basically the same as the previous method, but with more control over the media that gets loaded when events are prepared. With this method, only the media associated to both the events that are prepared and the game syncs that are currently active is loaded into memory.

Let's say you have a simple project with two events:

“Play_Maincharacter_FootSteps” and “Play_Monster_Footsteps”. Each event plays a different switch container that plays different random sounds based on the ground texture under the moving character. The switch group name is “GroundTexture” and has three possible states: “Snow”, “Concrete”, and “Sand”.

The switch container hierarchies in Wwise will look like the following:



In this example, we have 18 sounds (6 groups of 3 sounds) that can potentially be loaded into memory. You could use [Method 4: Preparing Action Events](#), but you won't get a level of granularity smaller than 6 sounds loaded in memory per event.

You could use [Method 3: Micromanaging Your Media](#) to get a better level of granularity, but you would have to create 6 different banks for this simple project (the number of banks increases quickly in a real project). Then, when a monster appears, you would have to check to see what textures are possible and then load the appropriate banks.

With the current method, all you have to do is specify which events and game syncs are possible and then only the appropriate media will be loaded. To make things very simple, all the media can be grouped into one single bank.

To setup banks when preparing events and game syncs:

1. Create a bank named “Events” and load it into the SoundBank Editor.
2. Drag the 2 events to the Add tab of the SoundBank Editor.
3. Deselect the “Media” check box, leaving the “Events” and "Structures" check box selected for both events. When using prepare event, the media must not reside in a bank, but be loaded directly off disk.



Note

For the purposes of this example, all the events and structures are contained in one bank. Although this is acceptable for small projects, you will most likely want to split up your content into several banks. It is also possible to create a separate “Structures” bank that does not need to be explicitly loaded nor prepared from the SDK command because each event includes references to the other banks that need to be loaded.

4. Generate the banks and copy the generated bank folder to the game application.



Note

The structure data contained within a single bank can't be split at run time. Therefore, if you are using `AK::SoundEngine::PrepareEvent`, and the structure data from a separate bank is required, all the structures within that bank will be loaded at once. For this reason, you may want to split the structure content in your project into multiple banks, to minimize the amount of unnecessary information that is loaded into memory.

To prepare events and game syncs in game:

1. Load the “Event” bank before the two events are required by the game.
2. Activate the Concrete texture at all times.
3. Prepare the main character footstep event at all times.
4. When possible ground textures are nearby, activate the game syncs.

The following sample of code gives you an idea how to prepare game syncs in game. First off, you need to initialize the sound engine and set the required settings, including the `bEnableGameSyncPreparation` flag.


```
// Initializing the sound engine.

AKInitSettings initSettings;
AKPlatformInitSettings platformInitSettings;
AK::SoundEngine::GetDefaultInitSettings( initSettings );
AK::SoundEngine::GetDefaultPlatformInitSettings( platformInitSettings );

// Set the required settings.

...

// Set PrepareEvent related settings.

// The flag bEnableGameSyncPreparation is set to true to activate
// the prepare gamesync mechanism. When set to false, the media
// associated with all game syncs is loaded and there is no need
// to call AK::SoundEngine::PrepareGameSyncs.
//
// When set to true, no media that is game sync dependent will be
// loaded unless the game sync is activated by calling AK::SoundEngine::PrepareGameSyncs
initSettings.bEnableGameSyncPreparation = true;

// (Optional) Allocate a memory pool into which prepared media will be loaded.
// If this is not done, memory will be allocated directly in the default memory pool.
{
    initSettings.uPrepareEventMemoryPoolID = AK.MemoryMgr.CreatePool( NULL, 4*1024*1024, 1024, AKMalloc );

    // (Optional) Give the memory pool a name. This can be very useful for profiling.
    AK::MemoryMgr::SetPoolName( initSettings.uPrepareEventMemoryPoolID, L"PrepareEventPool" );
}

AKRESULT eResult = AK.SoundEngine.Init( initSettings, platformInitSettings );
if( eResult != AK_Success )
{
    // Handle error.
}
}
```

Then you can load the Init bank and the events bank.

```
// Load Init and the event/structure bank.
AKBankID bankID; // Not used in this sample.
AKRESULT eResult = AK::SoundEngine::LoadBank( L"Init.bnk", AK_DEFAULT_POOL_ID, bankID );
if( eResult == AK_Success )
{
    eResult = AK::SoundEngine::LoadBank( L"Events.bnk", AK_DEFAULT_POOL_ID, bankID );
}

// ... At this point,
// the two events are loaded, but not prepared. No media is currently loaded.
```

Now you can prepare the required events and game syncs.

```
AkLpCtstr pNameArray[1];

// Prepare the main character footstep event.
pNameArray[0] = L"Play_Maincharacter_FootSteps";
eResult = AK::SoundEngine::PrepareEvent( Preparation_Load, pEventsNameArray, 1 ); // 1 is the array size

// ... At this point,
// one event has been prepared, but no media has been loaded yet.

// Now since concrete is always available in the game.
pNameArray[0] = L"Concrete";
eResult = AK::SoundEngine::PrepareGameSyncs( Preparation_Load, in_eType, "GroundTexture", pNameArray, 1 );

// ... At this point,
// the 3 sounds, Sound_Concrete_main_1, Sound_Concrete_main_2, and Sound_Concrete_main_3 are loaded.

// Now, let's say that the main character enters a land where there is snow.
pNameArray[0] = L"Snow";
eResult = AK::SoundEngine::PrepareGameSyncs( Preparation_Load, in_eType, "GroundTexture", pNameArray, 1 );

// ... At this point,
// 3 more sounds just got loaded : Sound_Snow_main_1, Sound_Snow_main_2 and Sound_Snow_main_3

// Then let's say that a Monster suddenly appears.
pNameArray[0] = L"Play_Monster_Footsteps";
eResult = AK::SoundEngine::PrepareEvent( Preparation_Load, pEventsNameArray, 1 ); // 1 is the array size

// ... At this point,
// 6 more sounds just got loaded ( Sound_Concrete_Monster_1.2.3 and Sound_Snow_Monster_1.2.3 )

// And now our player decides to run away from the monster, and the monster goes after him.
// They run so far that they arrive at a place where there is no snow anymore.

pNameArray[0] = L"Snow";
eResult = AK::SoundEngine::PrepareGameSyncs( Preparation_Unload, in_eType, "GroundTexture", pNameArray, 1 );

// ... At this point,
// The 6 sounds that were related to snow ( Sound_Snow_Monster_1.2.3 and Sound_Snow_main_1.2.3 ) are unloaded from memory.

...
```

Additional Notes on this Method

The following table lists the pros and cons of preparing game syncs.

Pros	Cons
Bank generation process is simple.	Potentially increases the number of reads and seeks on the disk as the media assets will be loaded one by one.
Level of granularity for media is very small.	
Maintain a low overall memory usage.	Less control over the total amount of memory used.
Easy to automate the process.	Activating a game sync can cause high streaming bandwidth when a lot of events are prepared that require new data to be loaded.
Only useful media gets loaded.	

SoundBanks Tips and Best Practices

Before defining the contents of your SoundBanks, you may want to review the following sections, which provide you with a series of tips and best practices that can help you better manage the SoundBanks in your game.

Updating the Initialization Bank

Wwise updates the initialization bank each time one or more SoundBanks are generated. This is to ensure that the two sets of banks are in sync. The Initialization bank contains all the general information about a project, including information on the bus hierarchy, and information on states, switches, and RTPCs. If changes are made to any of these elements within your project, you will have to update this bank in your game along with the newly generated SoundBanks. If, however, no changes are made to these project elements, you can keep the older version of the Init.bnk.

Initialization bank - A special bank that contains all the general information about a project, including information on the bus hierarchy, and information on states, switches, RTPCs, and Environmental effects. The Initialization bank is automatically created every time Wwise generates the SoundBanks. The Initialization bank is usually loaded once at the beginning of your game so that all the general project information is easily accessible during game play. It must be the first bank loaded when starting a game, or else the other banks may not load. By default, the Initialization bank is named “Init.bnk”.

SoundBanks and Memory

If you include long sounds in your SoundBank, it can take up a lot of the platform's memory. To avoid taking up extra memory, you should try to stream long sound or music files. For information on how to stream sounds and music objects, refer to [Streaming Sounds](#) and/or [Streaming Your Music](#).

Grouping Common Elements into SoundBanks

It is a good practice to group common elements together into one SoundBank. For example, all elements that will stay loaded throughout the game, like the menus, the main character, and so on, should be grouped into one SoundBank. For elements that will be loaded and unloaded as the game is being played, you should group them into logical units or building blocks that can be exchanged or swapped throughout the game.

Using Work Units and Folders

To avoid having to edit your SoundBanks each time changes are made to your project, you can re-create the way your SoundBanks are set up using folders and/or work units. Since Wwise maintains an active link between elements in a SoundBank and those in your project, once these folders have been added to the SoundBank you will never have to edit the SoundBanks again because they will get updated automatically.

Using Event IDs in SoundBanks

At the final stages of your project, it is a good practice to use event IDs instead of event strings because Event IDs can be verified more quickly because the sound engine doesn't have to hash the name before proceeding.

Generating Integrity Report

It is a good idea to generate the integrity report before generating your SoundBanks. The integrity report displays a list of errors in your project along with possible ways to resolve them. By resolving all your project errors before the SoundBanks are generated, you can minimize the problems with the audio or motion in your game.

Using Multi-Select in the SoundBank Editor

If you are working on the Add tab of the SoundBank Editor and you need to exclude the same type of project element for several items, you can select multiple items in the Hierarchy Inclusion list, and then deselect one of the check boxes. This will exclude the type for all selected items. You can use the multi-select to include a type for many items as well.

SoundBank Name Restrictions

To get around a platform's file system limit on filenames, you can package your SoundBanks into file packages. A file package is a self-contained unit, so it completely abstracts the platform's file system, which removes any limits the system may have on filenames.

Chapter 35. Managing File Packages

Overview	762
Working with File Packager Projects	762
Managing File Packages within a Project	765
Downloadable Content Overview	770
Generating File Packages	772
Using File Packager Arguments in the Command Line	773
File Packager Tips and Best Practices	777

Overview

The File Packager is a stand-alone utility that groups the SoundBanks, loose media, and/or streamed media files for a Wwise project into one or more file packages. A file package is a self-contained unit that abstracts a file system. This means that by using file packages, you can avoid some of the limitations of a platform's file system, including the limit on the length of filenames as well as the actual number of files. File packages can also help you better manage language versions as well as downloadable content that is made available post release.

All information about a Wwise project's SoundBanks and streamed media files can be retrieved by importing the SoundBanksInfo.xml file into the File Packager. The SoundBanksInfo.xml file is created automatically by Wwise each time the SoundBanks are generated.

By default, all files are added to the default.pck file package, which is created in the root of the SoundBank path directory for each platform. You can, however, create new packages, manually add files to different packages, and modify the location where the packages will be saved.

You can use the File Packager to create your file packages manually, or you can automate this process, by using a command line to run the File Packager as part of the SoundBank generation process. This command line can be defined at the project level or as a custom SoundBank user setting. For more information on using the command line to launch the File Packager post SoundBank generation, refer to the following sections:

- [Defining the SoundBank Settings for Your Project](#)
- [Defining Custom Attributes for Your SoundBanks](#)
- [Using File Packager Arguments in the Command Line](#)

Working with File Packager Projects

Each session created within the File Packager can be saved as a project so that you can easily save your work and return to it at any time. The File Packager project contains all information related to the packages created within a session, including the number of packages, file assignment settings, and the contents and file order of each package.

Each time you open the File Packager a new session is automatically created. You can either save this session as a new project or open a project that you worked on previously.

To open the File Packager:

1. From the Start menu, click **All Programs > Audiokinetic > Wwise 20???.? build ????** > **Tools > File Packager**.

The File Packager opens with a new session.

Related Topics

- [Importing SoundBank Information into a Project](#)
- [Opening Existing File Packager Projects](#)
- [Saving File Packager Projects](#)

Importing SoundBank Information into a Project

Before you can start creating and populating your packages, you need to import the SoundBank data that was created when you generated your SoundBanks in Wwise. All this data is stored in the SoundBanksInfo.xml file, which is located in the platform folder of the GeneratedSoundBanks directory. Wwise generates a different SoundBanksInfo.xml file for each platform.

Each time a project is opened, the File Packager automatically reads the SoundBanksInfo.xml file to populate the Files to package view. If changes are made in Wwise to the SoundBank and streamed media files, inconsistencies between to the two projects may result. If any files are missing, they will be marked as such and highlighted in red.

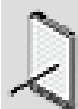
To import SoundBank information into a project:

1. Open the File Packager.
2. Click the **Browse** button beside the SoundBanks Info file field.

The Windows Open dialog box opens.

3. Navigate to the location of the SoundBanksInfo.xml file, select it, and click **Open**.

All SoundBanks and streamed files are loaded into the Files to package view.



Note

By default, all SoundBanks and streamed media files are assigned to the Default.pck package file.

Related Topics

- [Working with File Packager Projects](#)

- [Opening Existing File Packager Projects](#)
- [Saving File Packager Projects](#)

Opening Existing File Packager Projects

After a file packager project is saved, you can open it at a later time to continue working on it. Several instances of the File Packager can be open at one time, allowing you to work on more than one project simultaneously.

To open an existing File Packager project:

1. Open the File Packager.
2. From the menu bar, click **File > Open**.

The Windows Open dialog box opens.

3. Navigate to the folder where the File Packager project is located.
4. Select the .wfpproj file, and click **Open**.

The project information is loaded into the File Packager.

Related Topics

- [Working with File Packager Projects](#)
- [Importing SoundBank Information into a Project](#)
- [Saving File Packager Projects](#)

Saving File Packager Projects

Each session created within the File Packager can be saved as a project so that you can easily save your work mid-session and come back to it at any time. File Packager projects are saved as .wfpproj files.

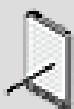
To save a File Packager project:

1. Do one of the following:

From the menu bar, click **File > Save**.

Press **Ctrl+S**.

All project information is saved as a .wfpproj file.



Note

If it is the first time you are saving a project, the Save As dialog box will open, prompting you to specify a name and location where you project will be saved.

Related Topics

- [Working with File Packager Projects](#)
- [Importing SoundBank Information into a Project](#)
- [Opening Existing File Packager Projects](#)

Managing File Packages within a Project

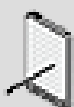
After the File Package project has been created, and the SoundBank and streamed file information has been imported into the project, you can start creating and populating the packages you will need. When this is complete, you can then generate the packages so they can be included on the game disk.

Managing the file packages in your project, consists of the following tasks:

- [Adding File Packages to a Project](#)
- [Removing File Packages from a Project](#)
- [Populating File Packages](#)
- [Ordering Files within a Package](#)

Adding File Packages to a Project

You can add as many packages to a project as required. After creating your packages, you can define the byte alignment of the packages using the Block size property. The number you specify will depend on the requirements of each platform's I/O device.



Note

For more information about block size and the constraints of each platform's I/O device, refer to the Streaming > Low-Level I/O section of [the SDK documentation](#).

To add file packages to a project:

1. From the Packages list, click **Add**.

A new packages is added to the Packages list.

2. Replace the default name with one that best represents the new package.
3. Press **Enter**.
4. Repeat steps 1-3, until all your packages have been created.
5. To specify the number of bytes upon which the data within your packages is aligned, enter a value in the Block Size field. The number you specify will depend on the requirements of each platform's I/O device. Typical values for each platform are as follows:

Xbox 360: 2,048

Wii, Nintendo 3DS and WiiU: 32

Other platforms: 1



Note

A block size of 1 means that there is no alignment.

Related Topics

- [Removing File Packages from a Project](#)
- [Populating File Packages](#)
- [Ordering Files within a Package](#)
- [Generating File Packages](#)

Removing File Packages from a Project

If you no longer need a package, you can easily remove it from your project at any time.

To remove file packages from a project:

1. From the Packages list, select the package that you want to remove from the project.
2. Click **Remove**.

The package is deleted from the project.

Related Topics

- [Adding File Packages to a Project](#)
- [Populating File Packages](#)
- [Ordering Files within a Package](#)
- [Generating File Packages](#)

Populating File Packages

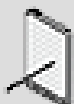
After you have created your packages, you can start populating them with SoundBanks and streamed media files.

You can populate the packages within your project in two different ways:

- [Assigning Files to a Package Automatically](#)
- [Manually Adding Files to a Package](#)

Assigning Files to a Package Automatically

Manually adding files to a package can be time consuming, so the File Packager allows you to create a set of rules that automatically assigns groups of files to specific packages based on language and/or file type. By setting these rules, you can greatly reduce the time it takes to populate your packages.



Note

The Default File Assignment tools only route files that have not been explicitly assigned to a package manually.

To route unassigned files to a package automatically:

1. In the Default file assignment view of the File Packager, assign the SoundBanks and/or streamed files for each language/SFX entry in the list to a specific package.
2. Assign all remaining SoundBanks to a specific package by selecting a package from the appropriate list.
3. Assign all remaining streamed files to a specific package by selecting a package from the appropriate list.

All files that have not explicitly been assigned to a package manually are automatically assigned to their appropriate package.

Related Topics

- [Manually Adding Files to a Package](#)
- [Adding File Packages to a Project](#)
- [Removing File Packages from a Project](#)
- [Ordering Files within a Package](#)
- [Generating File Packages](#)

Manually Adding Files to a Package

In some cases, you may want to manually add specific files to a package, for example, when creating new game content that will be available post-release. When files are manually added to a package, all automatic file assignments are overridden.

When you add a SoundBank to a package manually, you have the choice to include only the SoundBank, the streamed files associated with the SoundBank,

or both. At any time, you can switch to the Resulting Contents tab of the Package Contents view to review the actual contents of a package.

To add files to a package manually:

1. Select a package in the Packages view.

The contents of the package are displayed in the Package Contents view.

2. Drag one or more files from the Files to package view to the Added Items tab of the Package Contents view.

The number of files manually added to the package will appear in brackets on the Added Items tab title bar.

Also notice that the Resulting contents tab has a number displayed in the title bar. This number represents the total number of files included in the package. The two numbers may differ, for example, when a single SoundBank is added to the package that includes references to several streamed media files.

3. When you add a SoundBank to a package, you get to choose what type of files are actually added to the package by selecting one of the following options from the Inclusion mode list:

All files - Includes the SoundBank file and any associated streamed media files.

SoundBank only - Includes only the SoundBank file.

Streamed files - Includes only the associated streamed media files and not the actual SoundBank file that was added.

4. Switch to the Resulting Contents tab to review the actual contents of your package.

Related Topics

- [Assigning Files to a Package Automatically](#)
- [Adding File Packages to a Project](#)
- [Removing File Packages from a Project](#)
- [Ordering Files within a Package](#)
- [Generating File Packages](#)

Ordering Files within a Package

By default, all files within a package are added to the package in no particular order. You may, however, want certain files to be close to each other to minimize disk seeking during the game. You can arrange the files within a package using the File Order Editor.

Since ordering all files within a package can be time-consuming and unnecessary, the file packager allows you to arrange only those files that require ordering. All remaining files are automatically added to the ordered list using a placeholder, called 'Remaining files inserted here'. When you manually add a file to the ordered list, the file is automatically removed from the placeholder.

To order files within a package:

1. Do one of the following:

Double-click a package in the Packages list.

Select a package from the Packages list and click **Edit file order**.

The File Order Editor opens.

2. In the Package files view, select the file(s) that you want to manually arrange in a particular order.
3. Click the **Add to ordered list** button.

The file(s) are added to the next free position in the ordered list.



Tip

You can also drag files from the Package files view directly to a specific position in the ordered list.

4. If you want to re-order a file in the list, simply drag it to the new location in the ordered list.

A horizontal line displays the insertion point.

5. Continue to re-arrange the files until you have created a final order for the files within the package.



Tip

You can remove files from File order list, by selecting the file(s) and clicking **Remove from ordered list**.

Related Topics

- [Adding File Packages to a Project](#)
- [Removing File Packages from a Project](#)
- [Populating File Packages](#)
- [Generating File Packages](#)

Downloadable Content Overview

Downloadable content, or DLC, is typically a package of audio content added to the game after the main build (the initial game release without any downloadable content).

During the production phase of the game, Wwise produces SoundBanks and streamed files. The **File Packager** is used with the **Default file assignment** to have all SoundBanks and streamed files packaged into a single package, although they can be split by language or in streamed files, loose media, and Soundbanks.

When the game is released, however, the content used in the main release stops being packaged using the **Default file assignment#**. All content is manually inserted into the packages:

- Main packages are created manually
- All content is dragged to the main packages

Those packages, created manually, basically contain all SoundBanks and streamed files that were shipped in the main release. This is the point of reference for identifying newly created content. The **File Packager** project file is then saved.



Note

All DLC content must be created using the same Wwise Project that was used for the main release. Additionally, the same Wwise version must be used for both releases to ensure compatibility of the SoundBank and package formats.

In the File Packager

- For each DLC, a **File Packager** package is manually created.
- The user identifies new# content in the **File Packager**.
- The user drags the new content into the manually created package (for DLC):

Then, the content present in the main release can be modified. When an existing SoundBank or streamed file is modified, it must be repackaged into a new package. The game will then load both the main release package(s) and the new (DLC) package(s) by making sure they are loaded in chronological order. In loading different versions of the same object, Wwise gives priority to the most recent one.

Example

- SoundBankA was included in the main release and packaged into Package #1.

- The content of SoundBankA is modified after the main release. It is repackaged into Package #2, which is new, and becomes part of the DLC.
- The game, which has the DLC, loads Package #1 and then loads Package #2.
- When the game wants to load SoundBankA, Wwise will load the one found in Package #2 (it has priority over the one found in Package #1).

DLC Considerations and Limitations

In order to deliver the desired sound structure in a DLC, it is necessary to understand the impact of different changes and additions on the existing release structure. The following paragraphs detail common scenarios that require special considerations.

Adding children

To add children to some objects in a DLC you must completely repackage the existing SoundBank, which must be included in the DLC. However, some objects can have children added by a DLC via additional SoundBanks. The following table lists the objects that need a newly repackaged SoundBank and those that can add children in a DLC with an additional SoundBank.

Objects requiring repackaging of SoundBank to add children to DLC	Objects that use additional SoundBanks to add children to DLC
<ul style="list-style-type: none"> • Random/Sequence Container • Switch Container • Blend Container • Music Switch Container • Music Playlist Container • Music Segment 	<ul style="list-style-type: none"> • Actor-Mixer • Folder • Work Unit

For example, suppose the following hierarchy:

- WorkUnit
 - ActorMixer
 - ContainerA
 - Sound1
 - Sound2

Suppose you included ContainerA into SoundBank1 of the main release. For the DLC, you would be able to add a new container under the **Actor-Mixer Hierarchy** without issue.

- WorkUnit
 - ActorMixer
 - ContainerA
 - Sound1
 - Sound2

- ContainerB
 - Sound1
 - Sound2

You then create a new SoundBank2 and add ContainerB to it. When loading SoundBank1 and SoundBank2, the structures will be merged automatically. However, it would not be possible to add a sound under ContainerA for the DLC and package it inside a new SoundBank. If you want to do that, you must regenerate SoundBank1 and include it in the DLC. It will replace the main release version.

Modifying the Init bank

For the DLC, you must repackage the new version of the Init.bnk file in the DLC when you modify busses, game syncs (Game Parameter, Switches, and States), or environmental effects in any way.

Releasing multiple DLCs

When releasing multiple DLCs in parallel or in series, special attention must be taken. When content evolves over time, DLC builds must be built over the previous DLC projects.

For example, suppose you release the main release (include Init.bnk + RTM.bnk), the DLC-A (include Init.bnk + DLC-A-specific.bnk), and the DLC-B (include Init.bnk + DLC-B-specific.bnk). If the Bus structure is different in RTM, DLC-A, and DLC-B, then any additional busses found in DLC-A must be present inside DLC-B.

The 3 builds will need to include the Init.bnk file, which includes busses. Users could have:

- No DLC
- DLC-A only
- DLC-B only
- DLC-A and DLC-B

When users have DLC-A and DLC-B, it will load the Init.bnk provided in DLC-B. Since the Init.bnk in DLC-B was built over DLC-A, the busses required for DLC-A are available.

Generating File Packages

After the package(s) in your project are created and populated, and the file ordering is set, you can generate one or all packages. File packages are saved as .pck files in the platform folder of the Generated SoundBanks. You can, however, specify a different location, if required.

If any errors occur during generation, they are displayed in the generation log. If no errors occur, you can go ahead and add these package files to the game disk.

To generate file packages:

1. To specify a new location where the file packages will be saved, click the **Browse** button beside the Output directory field.

The Browse for Folder dialog box opens.

2. Navigate to the location where you want the file packages to be saved, select the folder, and click **OK**.

The full path is added to the Output directory field.

3. From the menu bar, click **Generate > All packages**.

The Generating Packages dialog box opens and displays the progress of the generation process. If errors are encountered, they will be highlighted in the log by a yellow circle.



Note

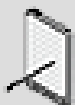
If you don't want to generate all packages each time, you can select one or more packages in the Packages view and then click **Generate > Selected Packages**.

Related Topics

- [Adding File Packages to a Project](#)
- [Removing File Packages from a Project](#)
- [Populating File Packages](#)
- [Ordering Files within a Package](#)
- [Using File Packager Arguments in the Command Line](#)

Using File Packager Arguments in the Command Line

When generating the package file from the command line, you must use specific arguments to define the characteristics of the package. The following table lists the arguments that can be used within the command line.







Note

The arguments used in the 2009.2 version of the File Packager and previous versions are compatible with the current version.

Argument	Description
-generate	<p>Launches the file package generation process. This argument can be used in combination with the file packager project path or with the -info argument. Here are some examples:</p> <p>Generate using a File Packager Project:</p> <pre>-generate <project_path></pre> <p>Generate a single package using a SoundbanksInfo.xml file:</p> <pre>-generate -info <soundbank_info_filename> -output <package_name></pre> <p>Generate a stream package and a bank package using a SoundbanksInfo.xml file:</p> <pre>-generate -info <soundbank_info_filename> -output_stm <package_name> -output_bnk <package_name></pre> <p>Generate a RSX stream package using a SoundbanksInfo.xml file:</p> <pre>-generate -info <soundbank_info_filename> -output_rsx <rsx_package_dir></pre>
-info <filename>	<p>Specifies the location of the SoundBanksInfo.xml file.</p> <p>Use with one or many of the following:</p> <pre>-output</pre> <pre>-put_</pre>
-output <filename>	<p>Generates one file package that include both SoundBank files and streamed media files. <filename> specifies the package name and location where the generated file package will be saved.</p> <p>This argument can be specified when the <projectpath> is not specified. It can either be a full file path or just the package name. Both options require the inclusion of the file package extension.</p> <p>This argument can be used in combination with -output_stm, -output_bnk or -output_rsx to generate more than one package file, where each one contains a different set of files.</p> <p>Although this value is defined in your project, if the -output argument is explicitly defined, it takes precedence over the value specified in the project.</p>
-output_stm <filename>	<p>Generates one file package that include only streamed media files. <filename> specifies the package name and location where the generated file package will be saved.</p> <p>This argument can be specified when the <projectpath> is not specified. It can either be a full file path or just the package name. Both options require the inclusion of the file package extension.</p>

Argument	Description
	<p>This argument can be used in combination with <code>-output</code>, <code>-output_bnk</code> or <code>-output_rsx</code> to generate more than one package file, where each one contains a different set of files.</p>
<p><code>-output_bnk <filename></code></p>	<p>Generates one file package that include only SoundBank files. <code><filename></code> specifies the package name and location where the generated file package will be saved.</p> <p>This argument can be specified when the <code><projectpath></code> is not specified. It can either be a full file path or just the package name. Both options require the inclusion of the file package extension.</p> <p>This argument can be used in combination with <code>-output</code>, <code>-output_stm</code> or <code>-output_rsx</code> to generate more than one package file, where each one contains a different set of files.</p>
<p><code>-output_rsx <directory></code> <code>-output_rsx</code></p>	<p>Generates one or many file packages that include only RSX streamed files. <code><directory></code> is optional and specifies the location where the generated file packages will be saved. If <code><directory></code> is not specified, the default directory is used, which is the <code>SoundBanksInfo.xml</code> directory.</p> <p>This argument can be specified when the <code><projectpath></code> is not specified. It can either be a relative or absolute directory. It can also be used in combination with <code>-output</code>, <code>-output_stm</code>, or <code>-output_bnk</code> to generate more than one package file, where each one contains a different set of files.</p>
<p><code>-output_loose <directory></code> <code>-output_loose</code></p>	<p>Generates one or many file packages that include only loose media files. These are files that are referenced by events or structures in at least one SoundBank, but where the media is not included in any SoundBank in the project. <code><directory></code> is optional and specifies the location where the generated file packages will be saved. If <code><directory></code> is not specified, the default directory is used, which is the <code>SoundBanksInfo.xml</code> directory.</p> <p>This argument can be specified when the <code><projectpath></code> is not specified. It can either be a relative or absolute directory.</p> <p>This argument can be used in combination with <code>-output</code>, <code>-output_stm</code> or <code>-output_bnk</code> to generate more than one package file, where each one contains a different set of files.</p>
<p><code>-blocksize <number></code></p>	<p>The number of bytes upon which the data within your package is aligned. The number you specify will depend on the requirements of the platform I/O device.</p> <p>Typical values for each platform are as follows:</p> <p>Xbox 360: 2048</p> <p>Wii, Nintendo 3DS and WiiU: 32</p> <p>Other platforms: 1</p> <p>The default value of 1 means that there is no alignment.</p> <p>For more information about block size and the constraints of each platform's I/O device, refer to the Streaming > Low-Level I/O section of the SDK documentation.</p>

Argument	Description
-blocksize_rsx <number>	<p>The number of bytes upon which the data within your package is aligned for RSX™ packages (PS3™ only).</p> <p>PlayStation 3: 128</p> <p>The default value is 128.</p>
-hideprogressui <boolean>	<p>Specifies whether the Generating Packages dialog is displayed while the packages are being generated from within Wwise.</p> <p>If set to true, the progress dialog is not displayed. By default, this argument is set to false.</p>
<projectpath>	<p>Specifies a File Packager project path that points directly to a .wfpproj file. When used with the -generate argument, other arguments are ignored and the File Packager project is used to generate the packages.</p> <p>This argument is optional.</p>
-soundbanks_dir <directory>	<p>Overrides the information in the SoundBanksInfo.xml file (at SoundBanksInfo/RootPaths/SourceFilesRoot) that specifies the directory where the generated SoundBanks were saved by Wwise.</p> <p>This is normally written as follows: 'projectpath\GeneratedSoundBanks\platform\'</p> <p>This argument is optional.</p>
-cache_dir <directory>	<p>Overrides the information in the SoundBanksInfo.xml file (at SoundBanksInfo/RootPaths/SourceFilesRoot) that specifies the directory where the converted files were saved by Wwise.</p> <p>This is normally written as follows: 'projectpath\cache\platform\'</p> <p>This argument is optional.</p>
-languages "<language1> <languageN>"	<p>Specifies the list of languages to package.</p> <div data-bbox="824 1283 1414 1562" style="background-color: #f0f0f0; padding: 10px; border: 1px solid #ccc;"> <p> Note</p> <p>The list is space-separated. Use double-quotes to enclose the list in one single argument.</p> <p>Use the keyword "SFX" in the language list to package them as well.</p> </div> <p>This argument is optional. By default all languages are packaged.</p> <p>This argument is only used when the File Packager project is not specified.</p>
-banks "<soundbank1> <soundbankN>"	<p>Specifies the list of SoundBanks to package. The names do not include the bnk extension.</p>

Argument	Description
	<div data-bbox="829 233 1417 428" style="background-color: #f0f0f0; padding: 5px;">  <p>Note</p> <p>The list is space-separated. Use double-quotes to enclose the list in one single argument.</p> </div> <p>This argument is optional. By default all SoundBanks are packaged.</p> <p>This argument can also be used when the File Packager project is specified. It will automatically build the packages that have a reference to the specified SoundBanks.</p>
<p>-includedFilesInBanks "<soundbank1> <soundbankN>"</p>	<p>Specifies a list of SoundBanks, from which the media files included in the bank, should be added as loose files to package. This option is only used if loose media is needed for use with the PrepareEvent API and this media is also needed in a bank loaded into memory. The names do not include the bnk extension.</p> <div data-bbox="829 827 1417 1022" style="background-color: #f0f0f0; padding: 5px;">  <p>Note</p> <p>The list is space-separated. Use double-quotes to enclose the list in one single argument.</p> </div> <p>This argument is optional. By default, media files included in SoundBanks are not added as loose files to packages.</p>
<p>-excludedFilesInBanks "<soundbank1> <soundbankN>"</p>	<p>Specifies a list of SoundBanks, from which the media files that are referenced but not included in the bank, should be added as loose files to package. This option can be used if loose media is needed for use with the PrepareEvent API and this media is also needed in a different SoundBank that is loaded into memory. The names do not include the bnk extension.</p> <div data-bbox="829 1310 1417 1505" style="background-color: #f0f0f0; padding: 5px;">  <p>Note</p> <p>The list is space-separated. Use double-quotes to enclose the list in one single argument.</p> </div> <p>This argument is optional. By default, media files that are referenced but not included in a SoundBank are added to the files package only if they are not referenced in any other SoundBank.</p>

File Packager Tips and Best Practices

Before defining the number and contents of your file packages, you may want to review the following section, which provides you with a series of tips and best practices that can help you make the most of the File Packager.

Packaging Content for Download Post-Release

It is becoming more and more common for companies to extend the life of a game by offering game players DLC (downloadable content) post-release. You can manage the deployment of this new content by following these simple steps:

To package the content for the final release:

1. In your Wwise project, generate the SoundBanks for the final build of your game.
2. Open the File Packager.
3. Import the final SoundBanksInfo.xml file generated by Wwise.
4. Set the default file assignments to **None**.
5. Manually add all files to a package.

This will make it easier for you to identify new content when it is added later on.

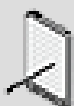
6. Generate the package(s) and then add them to the game disk.
7. Save the File Packager project to create a point of reference for the final release.

To package the DLC post-release:

1. When the new content is created in Wwise, generate the SoundBanks for the DLC.
2. Re-open the 'final release' File Packager project.

By default, the File Packager will scan the SoundBanksInfo.xml file to populate the Files to package view. Notice how all the new content that was added is not assigned to a package.

3. Create one or more packages for the downloadable content and then manually add the newly created SoundBanks and/or streamed media files to these packages.
4. Generate the new package(s) and then deploy them to your user base.
5. Save the File Packager project again to create a new point of reference for this DLC release.



Note

You can continue to deploy new content for your game in the same manner. Just remember to manually assign all files to a package and to save your File Packager project to create a point of reference for each release.

audio**kinetic**

Part VII. Working with Wwise



36. Getting to Know the Project Explorer	782
Overview	783
Visual Elements in the Project Explorer	785
Working in the Project Explorer	786
37. Getting to Know the Event Viewer	789
Overview	790
Working with the Event Viewer	791
38. Getting to Know the Property Editor	796
Overview	797
Working with the Property Editor	804
39. Getting to Know the Contents Editor	808
Overview	809
Working with the Contents Editor	816
40. Getting to Know the Transport Control	825
Overview	826
Setting Playback Properties	828
Pinning an Object in the Transport Control	831
Playing/Pausing/Stopping Content	832
Using Game Syncs During Playback	833
41. Getting to Know the Schematic View	838
Overview	839
Customizing the Schematic View	839
Working with the Schematic View	841
42. Getting to Know the Graph View	845
Overview	846
Changing the Display of the Graph View	847
Working with Control Points in the Graph View	852
Working with Curves in the Graph View	855
43. Getting to Know the Timeline	859
Overview	860
Working with the Timeline for Positioning	863
Working with the Music Segment Editor Timeline	864
44. Working with Searches, Queries, and References	866
Overview	867
Searching for Elements within Your Project	867
Finding the Project Elements that Reference a Particular Object	870
Working with Queries	872
Queries - Tips and Best Practices	880
45. Using Presets	882
Overview	883
Using Presets	883
46. Using a Control Surface	887
Overview	888
Connecting a Control Surface Device to Wwise	888
Creating a Control Surface Session	889

Understanding Control Surface Bindings	889
Creating Control Surface Bindings	890
Understanding Control Surface View Groups	893
Handling Conflicts in Control Surface Sessions	895
Using the Control Surface Toolbar	895

Chapter 36. Getting to Know the Project Explorer

Overview	783
Visual Elements in the Project Explorer	785
Working in the Project Explorer	786

Overview

The Project Explorer is where you manage all the elements in your Wwise project. In each of its tabs, you can create, rename, cut, copy, paste, and delete the distinctive elements that are displayed in the tab's hierarchical structure. Each tab also includes a toolbar that allows you to quickly add parent and child objects to your project hierarchy. In this one central place, you can organize and manage the various elements of your Wwise project, including audio, music, and motion assets, busses, events, SoundBanks, game syncs, and so on.

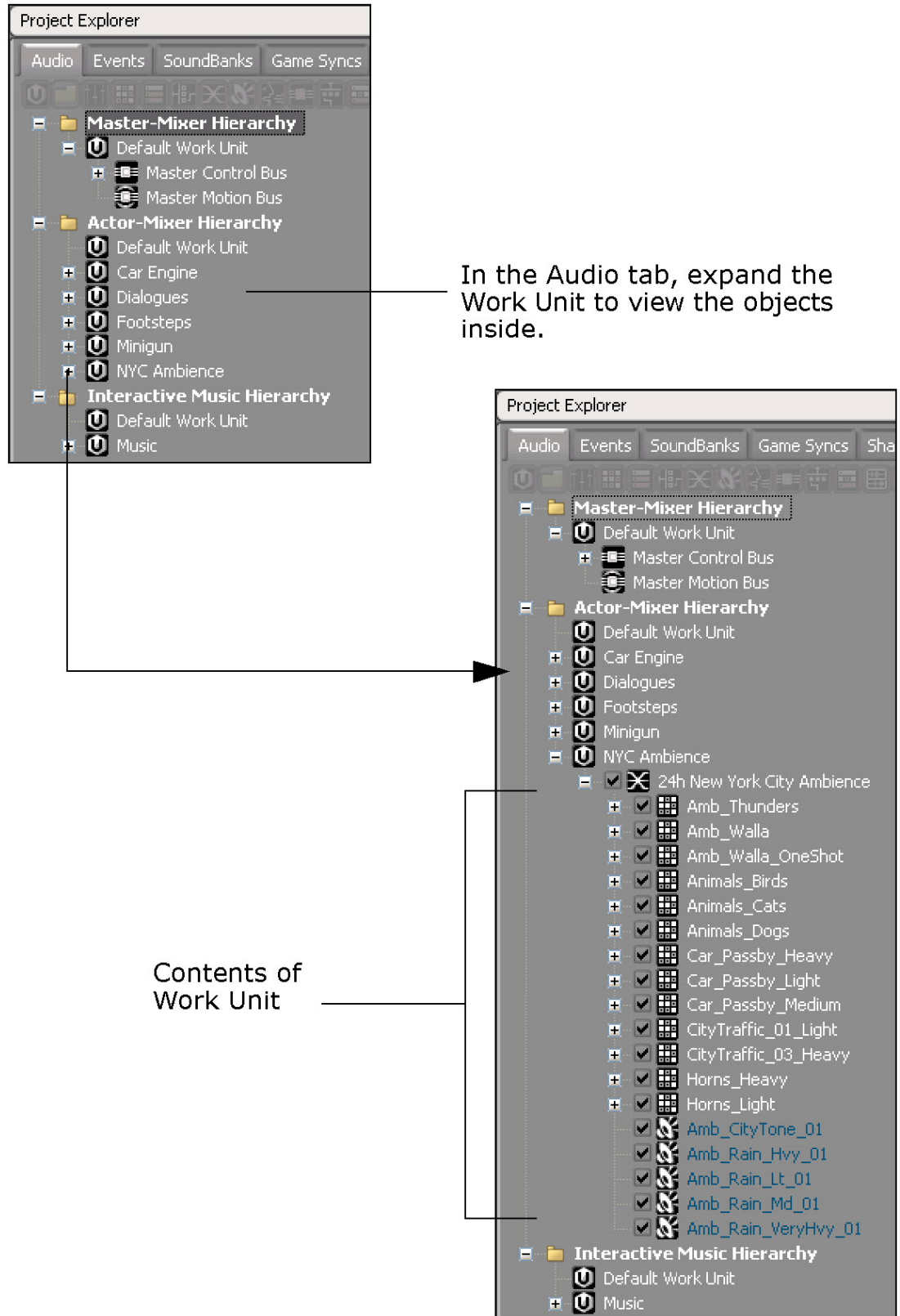
The Project Explorer is also where you can divide up your project elements into work units so that each member of your team can work on different parts of the project at the same time. For more information on creating work units, refer to [Dividing Your Project into Work Units](#).

The Project Explorer contains the following tabs:

- **Audio** - Contains all the sound, music, motion, and bus structures in your project.
- **Events** - Contains all the events, both action and dialogue events, in your project.
- **SoundBanks** - Contains all the SoundBanks in your project.
- **Game Syncs** - Contains all the states, switches, game parameters, and triggers in your project.
- **ShareSets** - Contains all the effect and attenuation ShareSets in your project.
- **Sessions** - Contains all the Soundcaster sessions in your project.
- **Queries** - Contains all the queries in your project.

To navigate through the different levels in the Project Explorer, you can expand and collapse the groups by clicking the plus (+) and minus (-) signs beside each object.






Getting to Know the Project Explorer



Visual Elements in the Project Explorer

Wwise uses a different icon to represent each object and element in your project to make it easy for you to identify the different object and element types within the Project Explorer and elsewhere in the interface. For a complete list and description of all icons, refer to [Understanding the Visual Elements in Wwise](#).

Other visual elements, such as color, are used in the Project Explorer to help identify the status of certain objects. For example, the color of an object can specify whether it has an associated source, or whether it has been converted for a particular platform. The following table contains a list of the visual elements used in the Project Explorer.

Visual Element	Example	Usage	Tab
Asterisk		Changes have been made to one or more project elements inside the work unit. When you save your project, the asterisk will disappear.	All
Check mark		When a check mark is displayed beside the object name, the object is included in the current platform. When no check mark is displayed, the object is not included in the current platform.	Audio
Object name in red		A sound, motion FX object, or music track has been created but is not associated with a source. A motion FX object will remain red until all enabled motion devices are associated with a source.	Audio
Object name in blue		A sound, motion FX object, or music track associated with a source that has not been converted for the current platform.	Audio
Object name in white		A sound, motion FX object, or music track is associated with a source that has been converted for the current platform.	Audio

If you are using a workgroup plug-in, special overlay icons will be displayed over the work units in the project explorer to help you identify the status of

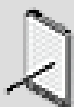
your project files. For example, different icons are used for files that are marked for add, checked out, and ones that are checked in. For more information about the overlay icons, refer to [Managing Project Files Using a Workgroup Plug-in](#).

Working in the Project Explorer

In the **Project Explorer** you can perform the standard Windows Explorer (or Mac Finder) commands such as renaming, cutting, copying, and pasting using the shortcut menu. You can also drag and drop project elements within the tabs as well as to other views within the Wwise interface. Keep in mind that whenever you move an object, you affect the parent child relationships. For information about building and managing these elements and their relationships, refer to the following sections:

- [Building the Actor-Mixer Hierarchy](#)
- [Chapter 22, *Building the Interactive Music Hierarchy*](#)
- [Chapter 14, *Managing Events*](#)
- [Chapter 15, *Managing Dynamic Dialogue*](#)
- [Chapter 34, *Managing SoundBanks*](#)
- [Working with Switches](#)
- [Chapter 16, *Working with States*](#)
- [Chapter 18, *Working with RTPCs*](#)
- [Chapter 19, *Working with Triggers*](#)
- [Chapter 20, *Working with States and State Groups for Dynamic Dialogue*](#)
- [Using Effects](#)
- [Managing Attenuation Instances](#)
- [Building a Simulation](#)
- [Creating a Query](#)

In addition to the standard commands, the Project Explorer shortcut menu includes tab-specific commands such as specifying platform inclusion and exclusion, importing and converting audio files in the Audio tab, and importing SoundBank definitions in the SoundBanks tab.



Note

You can have multiple instances of the Project Explorer open at the same time within the same layout.

Setting the Display Options

To help manage the complex hierarchies within your project, you might want to set the display of the Project Explorer a certain way so that you can quickly find

a particular project element when you need it. For example, you might want to automatically collapse every grouping so that you can navigate more easily in the Actor-Mixer hierarchy.

To change the Project Explorer display:

1. In any tab of the Project Explorer, right-click an object.

The shortcut menu is displayed.

2. From the shortcut menu, select **View Option**.

The View Option submenu is displayed.

3. Select one of the following options:

Auto Expand Child Object to expand all child objects when the parent object is expanded or collapsed. This option is disabled by default.

Expand All to expand all objects in the hierarchy.

Collapse All to collapse all objects in the hierarchy.

You can also use the following keyboard shortcuts to quickly navigate through the different levels in the Project Explorer.

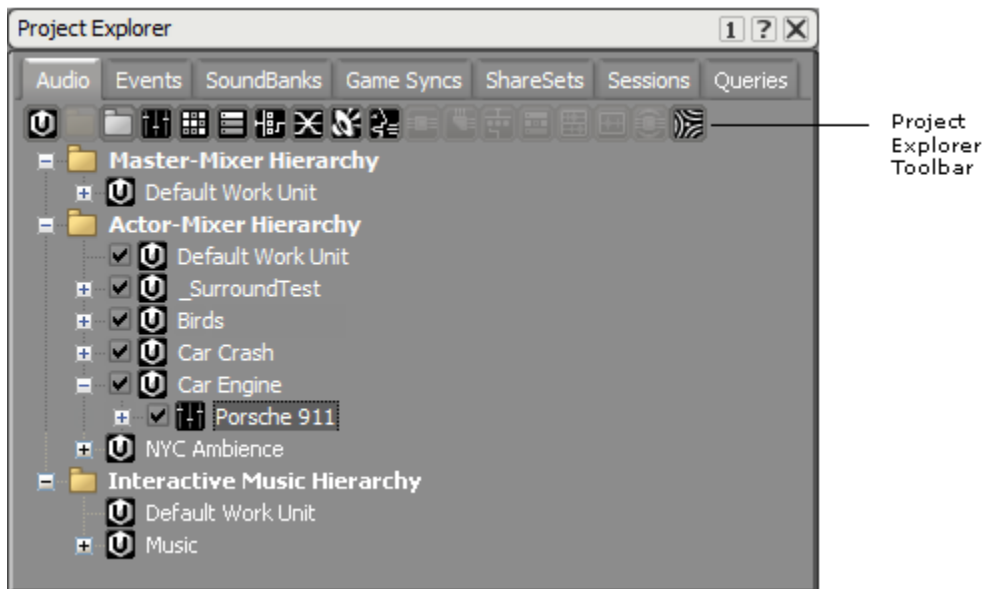
Keyboard Shortcut	To
Up arrow	Move up within the object list.
Down arrow	Move down within the object list.
Right arrow	Expand an object.
Left arrow	Collapse an object.

Related Topics

- [Using the Project Explorer Toolbar](#)
- [Visual Elements in the Project Explorer](#)

Using the Project Explorer Toolbar

Each tab of the Project Explorer contains a toolbar of icons. These icons represent the project elements that can be added as a parent or child of the selected node in the hierarchy. Different icons will become active depending on the type of object selected in the hierarchy.



By default, the toolbar displays all project elements that can be added as a child of the selected node. To display the possible parent project elements, press the Shift key.

To add a child object to the hierarchy using the toolbar:

1. In the Project Explorer, select an object or other project element in the hierarchy.

Icons representing the project elements that can be added as a child of the selected element become active in the toolbar.

2. Click one of the active icons to create a new child of the selected element.
3. Name the child object appropriately and then press **Enter**.

To add a parent object to the hierarchy using the toolbar:

1. In the Project Explorer, select an object or other project element in the hierarchy.
2. Press **Shift** to display the project elements that can be added as a parent of the selected element.
3. Click one of the active icons to create a new parent of the selected element.
4. Name the parent object appropriately and then press **Enter**.

Related Topics

- [Setting the Display Options](#)
- [Visual Elements in the Project Explorer](#)

Chapter 37. Getting to Know the Event Viewer

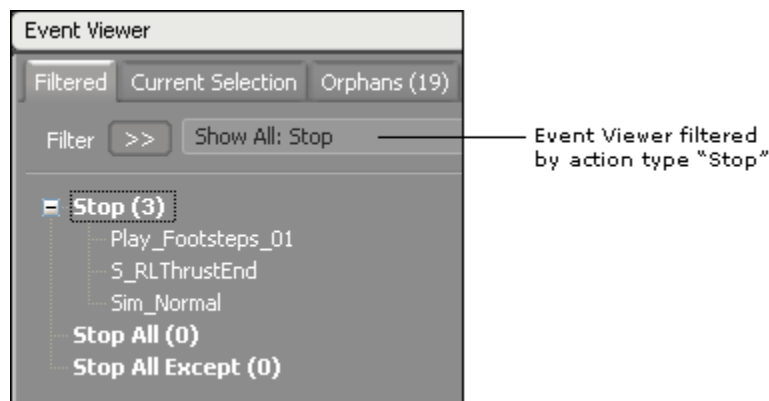
Overview	790
Working with the Event Viewer	791

Overview

Wwise uses “action” events to apply specific actions to the different sound, music, and motion structures within your project hierarchy. In a typical game, you could have hundreds of these events, so it is crucial for you to be able to find the ones you are looking for quickly. You can use the sorting and filter tools in the Event Viewer to find the different events that have been created for the current project.

The Event Viewer has three different tabs, each of which filters the events in a different way:

- **Filtered tab** - Displays all events alphabetically. You can also sort this list by action type using the Show All: Sorted option, or filter the list to display only those events that contain a particular action type. You can navigate through the filtered events by clicking the plus (+) and minus (-) signs to expand and collapse the folders.



- **Current Selection tab** - Displays a list of events that are associated with the object or objects that are currently selected in the Audio tab of the Project Explorer.
- **Orphans tab** - Displays orphaned events or events that have been created, but are not currently associated with a particular object.

You can also delete events, open the Event Editor, as well as drag and drop one or more events from the Event Viewer to other views within Wwise such as the Soundcaster or SoundBank Editor.



Note

Dialogue events are not displayed in the Event Viewer. Dialogue events can be viewed on the Events tab of the Project Explorer.

Working with the Event Viewer

The Event Viewer displays all the events that have been created in your project. Since you can have many events in one project, you will need to sort and filter the events to quickly find the ones you need to edit, play back, and so on.

The following sections will help you become familiar with the different tools and display options available in the Event Viewer.

- [Navigating the Event Viewer](#)
- [Sorting the Event List](#)
- [Filtering the Event List](#)



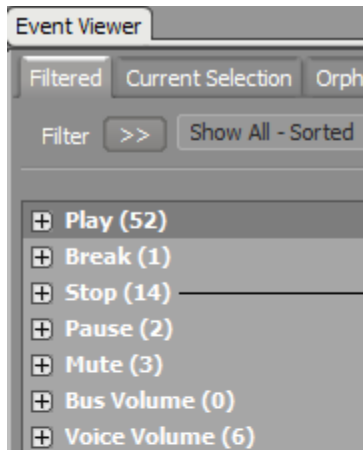
Note

You can have multiple instances of the Event Viewer open at the same time within the same layout.

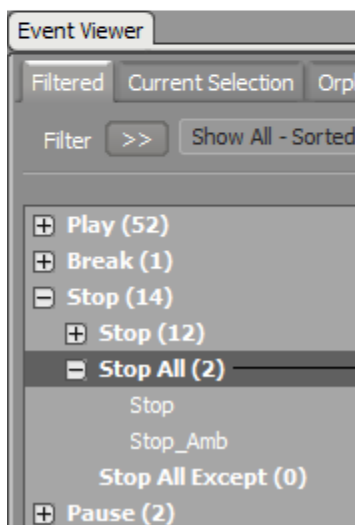
Navigating the Event Viewer

The information in the Event Viewer can be displayed in many different ways depending on how you have sorted or filtered the event list. Some filter and sorting options create levels within the Event Viewer. To navigate through the different levels, you can expand and collapse the groups by clicking the plus (+) and minus (-) signs beside each action category and type.

For example, to view a Stop action when Show All - Sorted is the Filter option, you must do the following:



Expand the Stop action category to display the different types of Stop actions



Expand the Stop action type grouping to display the different events that contain Stop actions

Quick Navigation

You can also use the following keyboard shortcuts to quickly navigate through the different levels in the Event Viewer.

Keyboard Shortcut	To
Up arrow	Move up within the event list.
Down arrow	Move down within the event list.
Right arrow	Expand an event action category or type grouping.
Left arrow	Collapse an event action category or type grouping.

Changing the Event Viewer Display

Depending on the situation or your own personal preference, you may want to change the display of the Event Viewer. For example, you can expand or collapse all groupings, or have the Event Viewer automatically expand every grouping each time the list is filtered or sorted.

To change the Event Viewer display:

1. Right-click one of the group headings in the Event Viewer.

A shortcut menu appears.

2. Select one of the following options:

Auto Expand to automatically expand every grouping each time the list is filtered or sorted.

Expand All to expand all category and type groupings.

Collapse All to collapse all category and type groupings.

Related Topics

- [Sorting the Event List](#)
- [Filtering the Event List](#)

Sorting the Event List

By default, the Event Viewer displays the Filtered tab where all the events in your project are listed in alphabetical order. To find specific events more quickly, you can select the Show All - Sorted filter, which sorts events by event action. This means that all the play actions are grouped together, all the stop actions are grouped together, and so on.

To sort the event list:

1. In the Event Viewer, click the **Filtered** tab.

By default, all events are listed alphabetically.

2. Click the **Filter** button to display the filter and sorting options.
3. Click the **Show All - Sorted** option.

All the events in your project are sorted by action category and then action type.

Related Topics

- [Filtering the Event List](#)
- [Navigating the Event Viewer](#)
- [Changing the Event Viewer Display](#)

Filtering the Event List

You can filter the event list to further refine your search for specific events. You can filter the events by:

- [Filtering the List by Action Type](#)
- [Filtering the List by Current Selection](#)
- [Filtering the List by Orphaned Events](#)

Filtering the List by Action Type

You can filter the event list by action type. For example, you can filter the list to show only the Stop All actions, or only the Mute actions.

To filter the event list by action type:

1. In the Event Viewer, click the **Filtered** tab.

By default, all events are listed alphabetically.

2. Click the **Filter** button to display the filter and sorting options.
3. Select an action category and then click an action type from the list.

The events are filtered according to the filter option you selected.

Related Topics

- [Filtering the List by Current Selection](#)
- [Filtering the List by Orphaned Events](#)
- [Navigating the Event Viewer](#)
- [Sorting the Event List](#)
- [Changing the Event Viewer Display](#)

Filtering the List by Current Selection

You can filter the list to find specific events that are associated with one or more objects selected in the Audio tab of the Project Explorer.

To filter the event list by current selection:

1. In the Audio tab of the Project Explorer, select one or more objects.
2. In the Event Viewer, click the **Current Selection** tab.

The events associated with the selected objects are displayed in the events list.



Note

If more than one object is selected in the Audio tab of the Project Explorer, the events are sorted by object.

Related Topics

- [Filtering the List by Action Type](#)
- [Filtering the List by Orphaned Events](#)
- [Navigating the Event Viewer](#)
- [Sorting the Event List](#)
- [Changing the Event Viewer Display](#)

Filtering the List by Orphaned Events

You can filter the list to find orphaned events, which are events that contain actions that are not associated with any object. The title of the Orphans tab contains a number in brackets that specifies the total number of orphaned events in your project.

To filter the event list by orphaned events:

1. In the Event Viewer, click the **Orphans** tab.

The orphaned events are displayed in the events list.

Related Topics

- [Filtering the List by Action Type](#)
- [Filtering the List by Current Selection](#)
- [Navigating the Event Viewer](#)
- [Sorting the Event List](#)
- [Changing the Event Viewer Display](#)

Chapter 38. Getting to Know the Property Editor

Overview	797
Working with the Property Editor	804

Overview

The Property Editor is a collection of properties and behavior options that you can use to define the overall characteristics of a particular object or game sync in your project.

When an object is loaded into the Property Editor it will contain some of the following tabs:

- [General Settings](#)
- [Effects](#)
- [Positioning](#) (for Actor-Mixer and Interactive Music Hierarchy objects only)
- [Real-Time Parameter Controls](#) (Real-time Parameter Controls)
- [States](#)
- [Transitions](#) (for Music objects only)
- [Stingers](#) (for Music objects only)
- [Advanced Settings](#)

When a game sync is loaded into the Property Editor, different properties will be displayed depending on the type of game sync you are editing. The following table describes which properties are displayed for each game sync:

Game Sync	Property Editor Contents
State Group	Transition settings between states.
State	Pitch, low-pass filter, high-pass filter and volume controls.
Switch Group	Switch to game parameter value mappings.
Switch	Name and notes
Game Parameter	Minimum and maximum values for the game parameter.
Trigger	Name and notes



Note

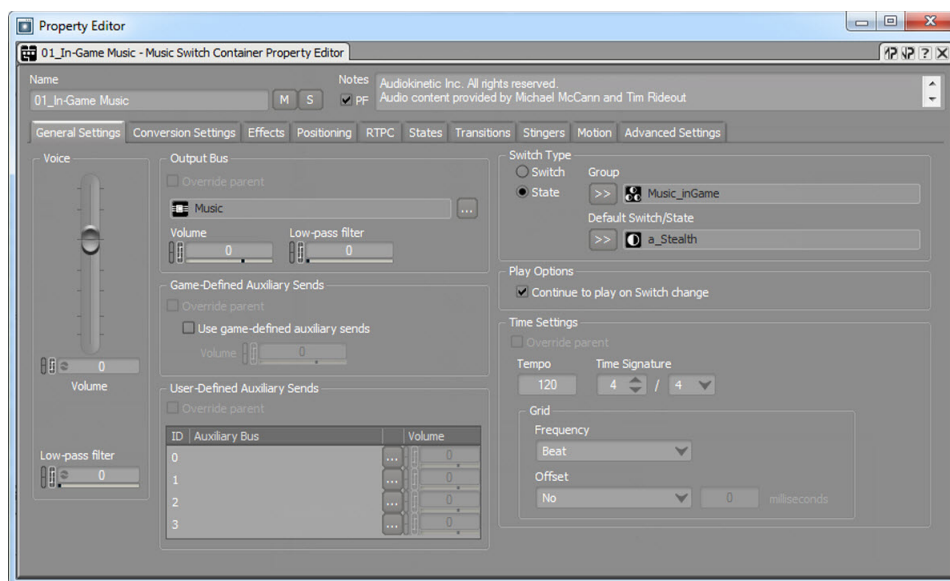
Since folders don't have any associated properties or behaviors, when they are loaded into the Property Editor, you will only be able to edit the folder's name and corresponding notes.

General Settings

The General Settings tab is the main tab within the object Property Editor. It contains a series of options where you can view and define the general

characteristics of objects within your project hierarchy. For example, you can set specific values for the volume, and pitch of an object. You can also define specific playback behaviors for the different objects within your hierarchy.

The General Settings tab is split between properties and behaviors. The properties appear on the left side of the tab and the behaviors on the right side. The properties are further divided into two groups: absolute and relative. The absolute properties are displayed on the left and the relative or cumulative properties on the right.



Properties and routing

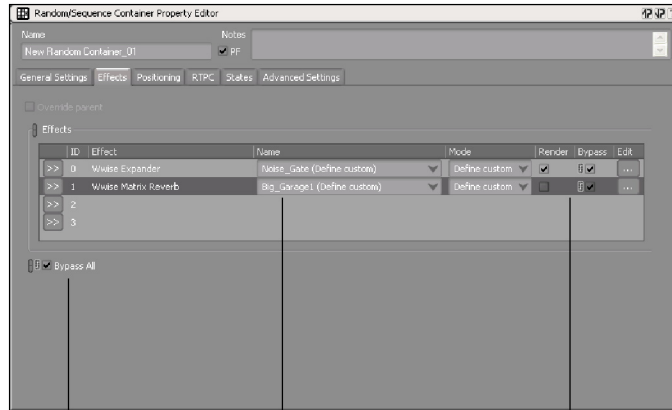
Behaviors

For a complete description of relative and absolute properties, refer to [About Properties in the Project Hierarchy](#).

The General Settings tab is also contextual, which means that the properties and behavior options that are displayed on this tab, will be different depending on the type of object you are currently editing.

Effects

The Effects tab is where you apply one or more effects to the objects or busses in your project hierarchy. When applying an effect, you must decide whether to apply a ShareSet or custom instance of the effect. You can also bypass, render, or edit the effect.



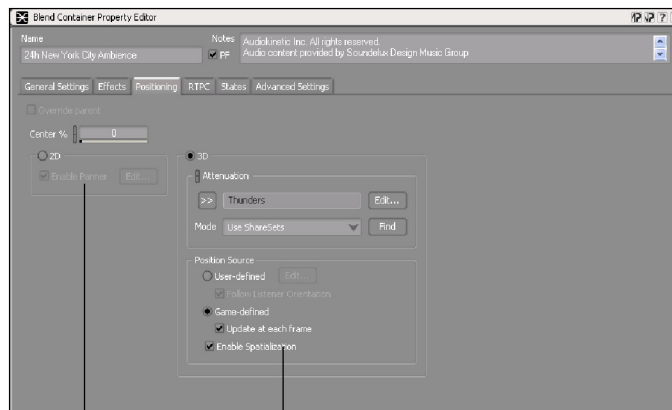
Bypass all effects at once

Apply multiple effects to selected object

Render or bypass individual effects

Positioning

The Positioning tab contains a series of options where you can view and define the positioning characteristics of project objects. You can set whether an object uses 2D or 3D positioning and then define how each of these types of positioning will be implemented. For sounds, you can also specify what percentage will be played through the center speaker.

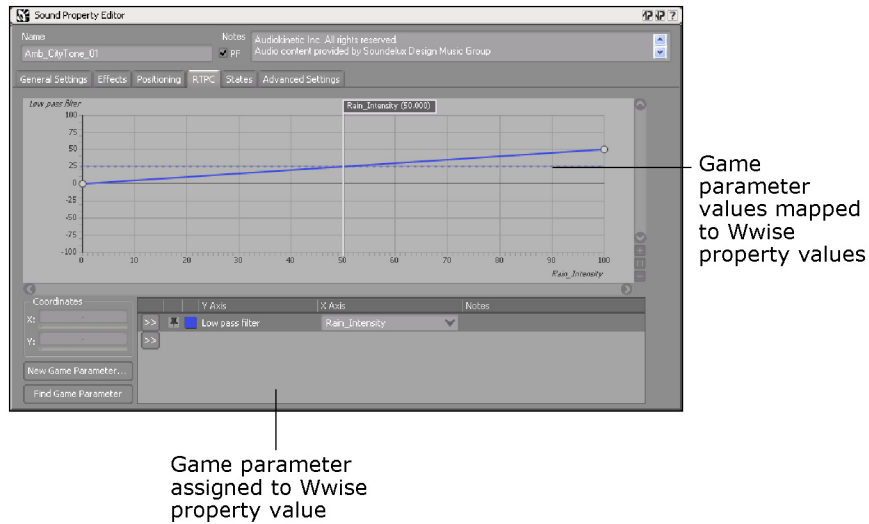


2D positioning options

3D positioning options

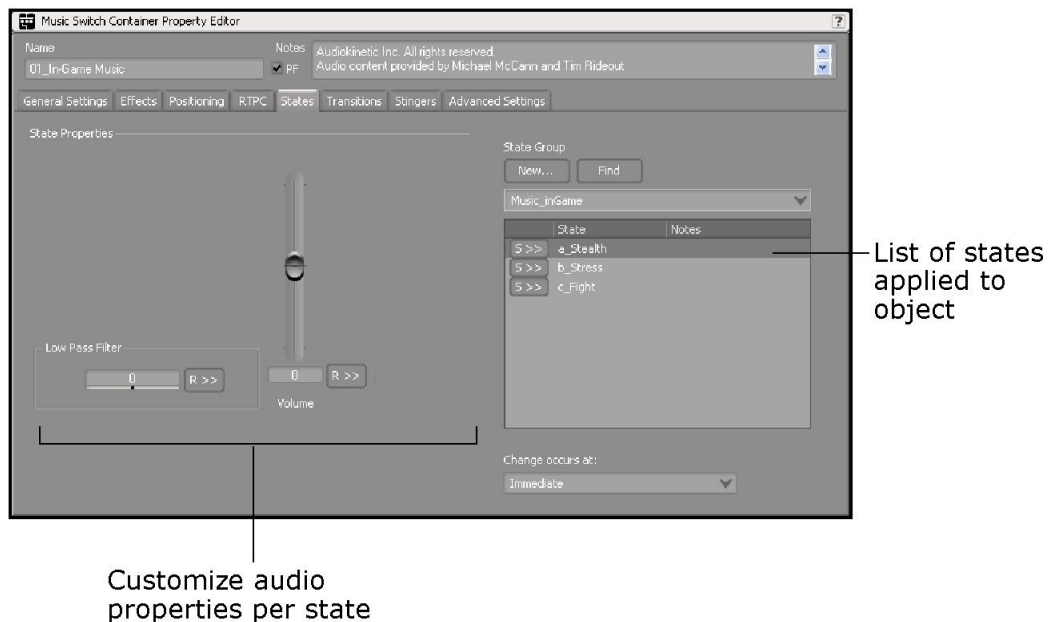
Real-Time Parameter Controls

The controls on the RTPC (Real-time Parameter Controls) tab enable you to edit specific sound properties in real time based on real-time parameter value changes that occur within the game. The parameter values are displayed in a graph view, where one axis represents the property values in Wwise and the other axis represents the in-game parameter values. You can focus on one curve at a time or edit all curves simultaneously.



States

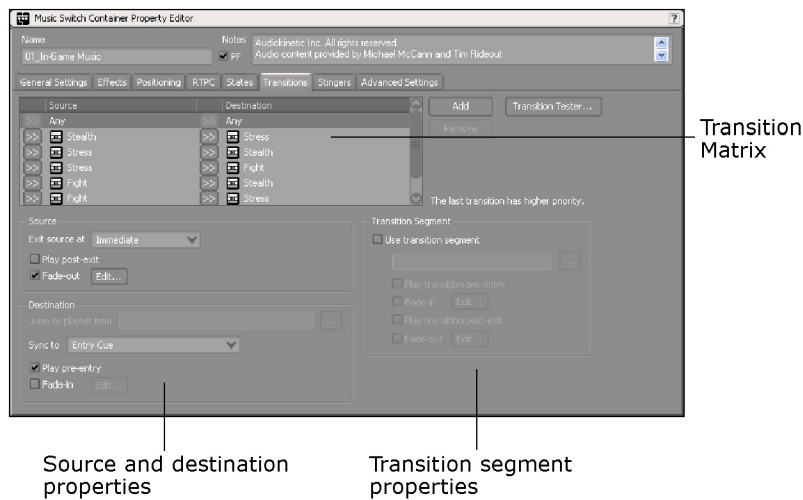
The States tab contains a list of possible states that can be assigned to the current object. You can subscribe a particular object to a state group and then customize certain audio properties, such as pitch and volume, for each of its states to further define the characteristics of the object.



Transitions

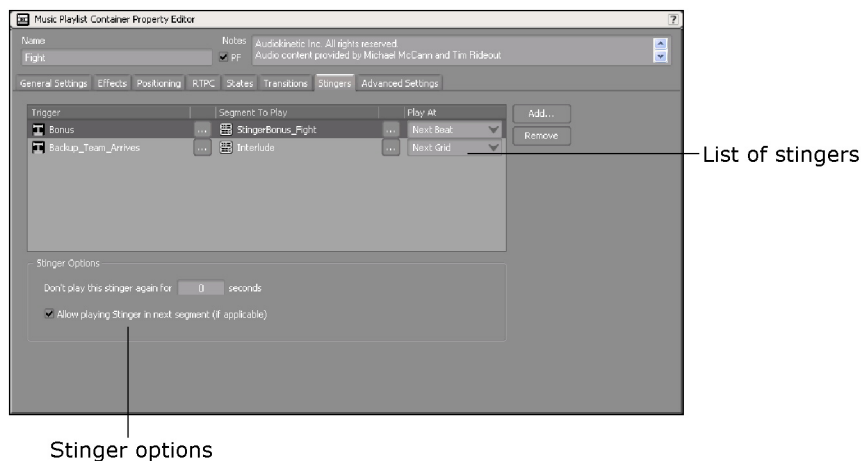
The main part of the Transitions tab is the Transition Matrix, a list of rules that defines how each object within a music switch or playlist container transitions to every other object within the container. It also contains controls

for customizing the properties of each source and destination in a transition, and for specifying the use and properties of transition segments.



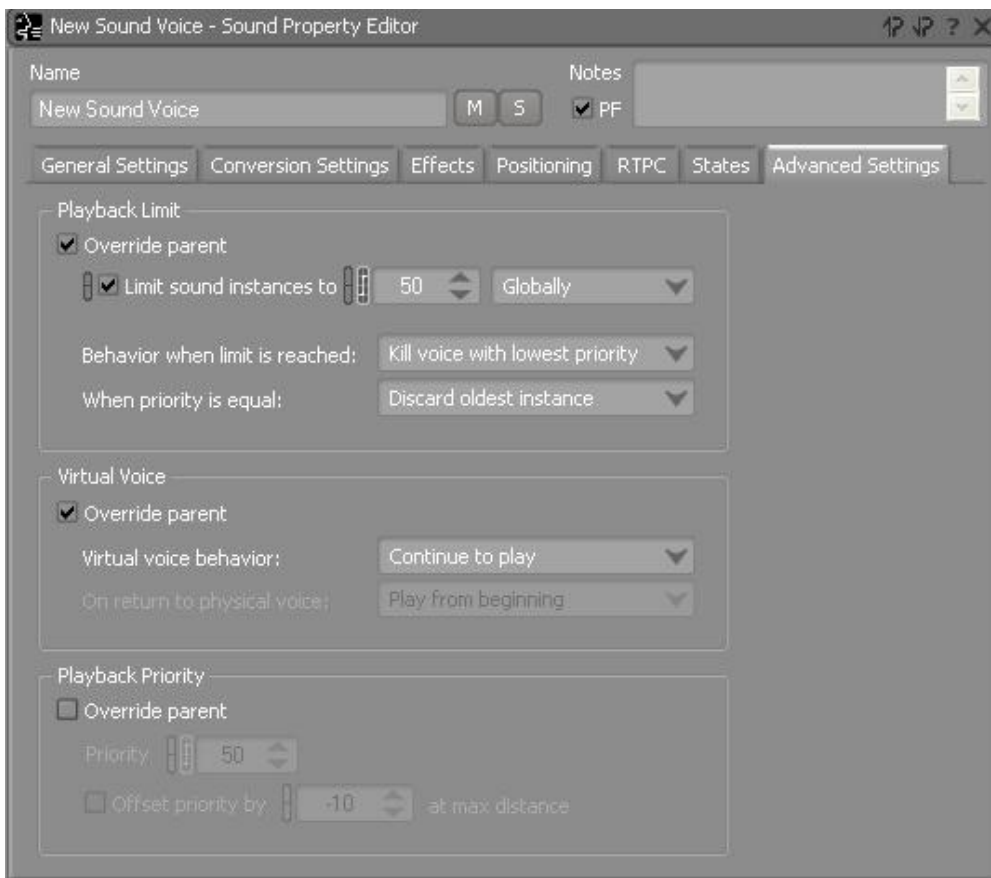
Stingers

In the Stingers tab, you can create a list of triggers, assign segments to play for them, and specify the time at which they should be played. You can also set play options for each stinger.



Advanced Settings

The Advanced Settings tab contains a series of advanced behavior options that can help you effectively manage the number of sounds, music, or motion objects that are played simultaneously in game, which ones take priority, and what happens when the volume of an object falls below a pre-defined volume threshold.



Using Advanced Settings and Dynamic Mixing

Sound structures' advanced settings are specified in the **Advanced Settings** tab of the object **Property Editor** to limit the number of sounds playing concurrently and to specify their behavior when they are inaudible, which respectively provide the following advantages:

- Performance improvements and help in cleaning up your mix
- Memory and CPU savings

Advanced settings and mixing

You shouldn't wait too late in the production process to adjust the advanced settings. More precisely, you should at least perform a first pass before or while mixing. If you spend time tweaking your mix too early, and later realize that audio is using too many resources, you might end up using advanced settings so aggressively to keep audio processing within bounds that it would have a dramatic impact on your mix.

Playback limits should actually help you with mixing. Use them as a form of dynamic mixing, to help players focus on what's important instead of drowning them with sounds. You may also use bus ducking, **Set Volume** actions, states, or RTPCs to clean up your mix.

Playback limit, priority, and under Volume Threshold behavior

Playback limits used on sound structures help you limit the number of sounds playing at the same time, per game object (if specified in the **Actor-Mixer** and **Interactive Music** hierarchies) or globally (if specified in the **Master-Mixer Hierarchy** - busses). Its logic is solely based on the number of sounds that play. Playback limiting conditions are checked before attempting to play a sound. When a sound is about to start and the limit has already been reached, either this sound or another one is stopped. The first criterion is the sound's priority. In the case where the two candidates have the same priority, the sound engine stops either the oldest or the newest instance of the sound, as specified by the **When limit is reached** and **When priority is equal** properties.



Note

When a sound is killed because of playback limit, it is not restarted when the play count falls below the limit. So be careful with infinitely looping ambient sounds.

Priority settings work together with the playback limit. Tweak priorities in the hierarchy to balance the limiting system. Sounds that should never be killed by the limiting system, like voice overs, background music or looping ambient sounds, should have the highest priority. Furthermore, the effective priority can be affected by the distance between the sound and the listener.

The "under **Volume Threshold**" behavior has nothing to do with playback limits and priorities. It only tells what will be the sound's behavior when it is inaudible. To determine whether a sound is inaudible, Wwise only looks at the metadata volume, that is, the resulting contribution of all volumes (and LFE volume) of the hierarchy, busses, states, RTPC, and Set Volume actions. It never analyzes WAV data.

Dynamic mixing techniques

The following dynamic mixing techniques can be applied in the **Advanced Settings** tab.

- **Limit playback on busses or on actor-mixer structures to make room for important sounds**

For example, reduce the number of ambient and foley sounds when there's a lot of action, explosions, or any element on which players should focus. Find the bus where ambiance and foley meet guns and explosions, set a limit on that bus, and lower the priority of the former.

- **Use distance offset for priority**

For example, among ambiance sounds, use playback limiting along with distance-based priority in order to focus on those that are closer. The **Offset priority by** option specifies a priority offset value at maximum distance, which is interpolated between 0 and **at max distance**.

- **Duck volume of less important sounds**

It is sometimes impossible to use playback limits. For example, if you start infinitely looping ambient sounds once at the beginning of a level, you should avoid having them stopped by playback limiting because they will not be restarted when things calm down. In this case, or in any case you see fit, use the technique you want to duck their volume when there is activity in other, more important areas of the game's audio. For example, shut ambiences down during voice overs or combat. You don't need to hear the buzz of a lamp when a grenade explodes next to you. This process can be viewed as metadata side-chaining. Volume changes can be triggered using the bus ducking feature on control busses or **Set Volume** actions within specific events, and so on.

Once you have cleaned up your mix by lowering the volume of less important sounds in specific situations, tweak their "under **Volume Threshold**" behavior in order to have them use the least CPU and memory possible while they are inaudible.

- **Implement a code-side dynamic mixing system**








Notice that the playback limits, priorities and priority offsets can be exposed to the game through RTPC. Nothing prevents you from implementing a system which adapts these settings based on what's happening in the game.

Working with the Property Editor

The Property Editor may contain a series of options, fields, sliders, lists, buttons, and a graph view that you can use to define the properties and behaviors of the different objects and game syncs within your project. If you require help using these tools, refer to the following sections:

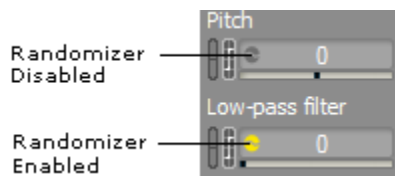
- [Using Text Boxes](#)
- [Using Lists](#)
- [Using Sliders](#)
- [Chapter 42, *Getting to Know the Graph View*](#)

When modifying a property value in the Property Editor, you may have the option of applying a randomizer and/or linking or unlinking the value across platforms. These features are represented in Wwise by a unique icon. Wwise also indicates whether a property value is assigned to a game parameter using an RTPC. The different icons are described in the following table.

Icon	Name	Description
	Link	The property value is linked to the values of other active game platforms.
	Unlink	The property value is not linked to the values of the other active game platforms.
	Partial Unlink	The property value for the current platform is linked to the other active platforms, but one or more corresponding values of other platforms are unlinked.
	RTPC - Disabled	This property value is not tied to an in-game parameter value.
	RTPC - Enabled	An in-game parameter value is tied to this property value. This means, for example, that the speed of a car in-game can be tied directly to the pitch property in Wwise. As the speed of the car increases in-game, the pitch in Wwise will increase in real time.
	Randomizer - Enabled	A property value to which a Randomizer effect has been applied.
	Randomizer - Disabled	A property value to which no Randomizer effect has been applied.

Randomizing Property Values

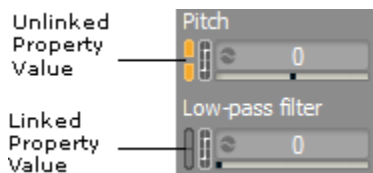
To enhance the realism of the sounds within the game, you can apply Randomizers to most of the property values within the Property Editor. The Randomizer specifies a possible range of values that can be used for a particular property. Each time the object is played, Wwise selects a different property value within the specified range. Applying Randomizers to each of the different property values will ensure that the object will sound differently each time it is played.



For more information on Randomizers and how to use them, refer to [Enhancing Sound and Motion by Randomizing Property Values](#).

Linking/Unlinking Property Values

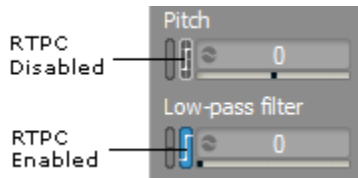
Wwise also allows you to simultaneously author on several platforms by linking and unlinking most of the property values associated with an object. Linking property values across active platforms sets all property values to the same value. Unlinking a property value allows you to customize it for a particular platform.



For more information on linking and unlinking property values, refer to [Customizing Object Properties per Platform](#).

Assigning Property Values to Game Parameters

As previously mentioned, you can assign game parameters to specific property values in Wwise. These are called real-time parameter controls (RTPCs). Wwise gives you visual feedback on which properties have been assigned as a RTPC. When a property value is assigned to a game parameter, the property's RTPC indicator becomes highlighted in blue.



For more information on using RTPCs, refer to [Controlling Property Values Using Game Parameters](#).

Displaying a Project Element's Properties

The Property Editor displays all the properties associated with a particular project element within your project. You can load a particular object into the Property Editor by double-clicking the object in a number of different views, such as the Audio tab of the Project Explorer, the Contents Editor, and the Capture Log. You can load a game sync into the Property Editor by double-clicking any one of the following in the Game Syncs tab of the Project Explorer:

- Switch group
- Switch
- State group
- State
- Game parameter
- Trigger
- Argument
- Argument value

To display a project element's properties from the Designer layout:

1. In the Designer layout, do one of the following:

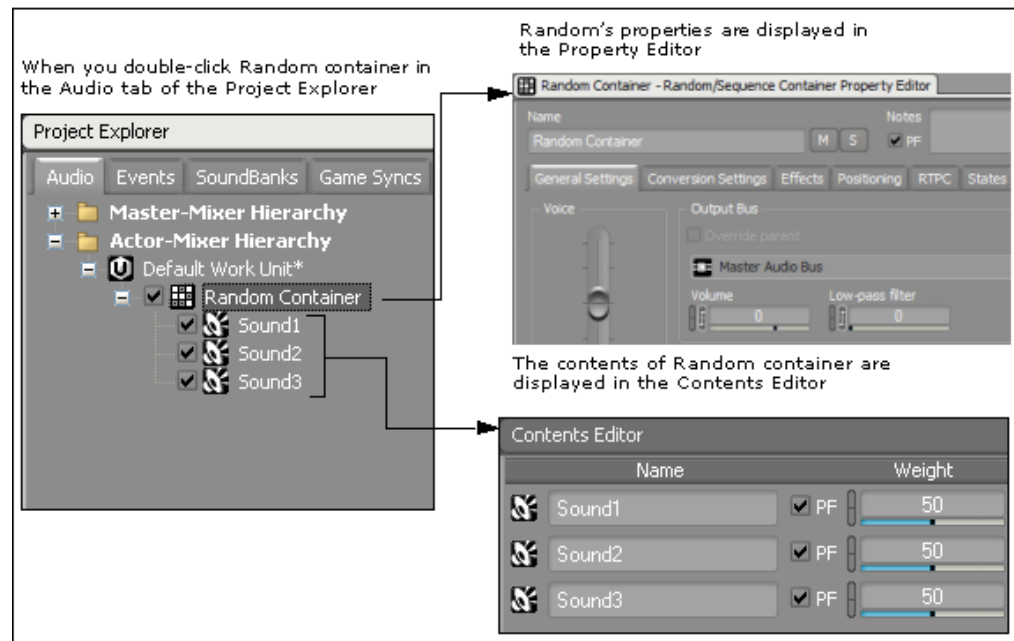
Double-click an object in the Audio tab of the Project Explorer.

Double-click a game sync in the Game Syncs tab of the Project Explorer.

The following occurs:

The object's properties are displayed in the Property Editor.

All child objects contained within the object or game sync are displayed in the Contents Editor.



Chapter 39. Getting to Know the Contents Editor

Overview	809
Working with the Contents Editor	816

Overview

The Contents Editor displays the object or objects that are contained within the parent object that is loaded into the Property Editor. Since the Property Editor can contain different kinds of object structures and other project elements, the Contents Editor handles them contextually which means that different layouts are displayed based on the type of object loaded.

When sound, music, or motion structures and other project elements are loaded into the Contents Editor, you have quick access to some of the most common properties associated with each object, such as volume. By having the settings in the Contents Editor, you can edit a parent's child objects without having to load them individually into the Property Editor. The Contents Editor also provides you with the tools to include or exclude objects from platforms, define playlists and switch behaviors, as well as manage audio and motion sources and source plug-ins.

The different views for the sound, music, and motion structures and other project elements in the Contents Editor are described in the following sections.

- [Sound Objects](#)
- [Motion FX Objects](#)
- [Random Container](#)
- [Sequence Container](#)
- [Switch Container](#)
- [Blend Container](#)
- [Actor-Mixer](#)
- [Folder](#)
- [Bus](#)
- [Event](#)
- [Switch Group](#)
- [State Group](#)
- [Music Track](#)
- [Music Segment](#)
- [Music Playlist Container](#)
- [Music Switch Container](#)

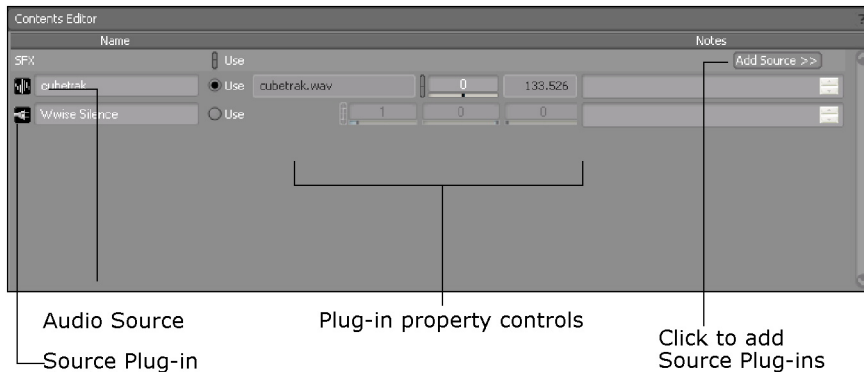
Sound Objects

When you load a sound object into the Property Editor, its sources will be displayed in the Contents Editor. A sound object can contain several sources including the following:

- **Takes** - Different versions of the same sound object that you can audition and test before choosing the source that you will use. These audio sources can link to audio files, silences, plug-ins or a combination of the three.

- **Languages** - Different language versions for the localization of your project. For information about how to work with language versions in Wwise, refer to [Localizing Your Project](#).

The controls and properties for each type of source in the Contents Editor will be different depending on the type of source that is selected.

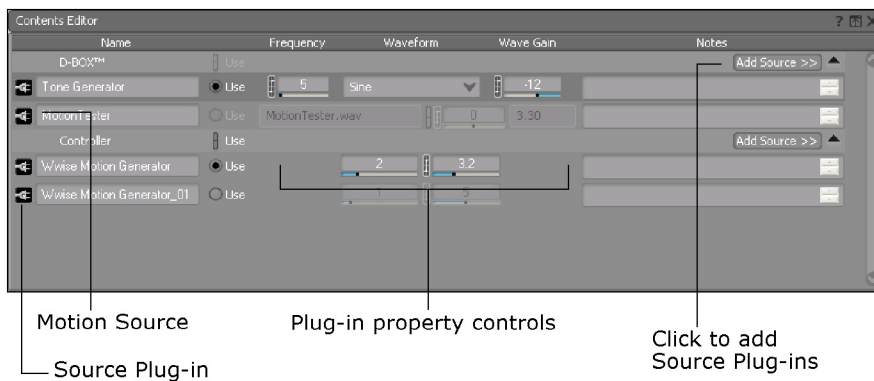


Motion FX Objects

When you load a motion FX object into the Property Editor, its sources will be displayed in the Contents Editor. A motion FX object can contain several sources including the following:

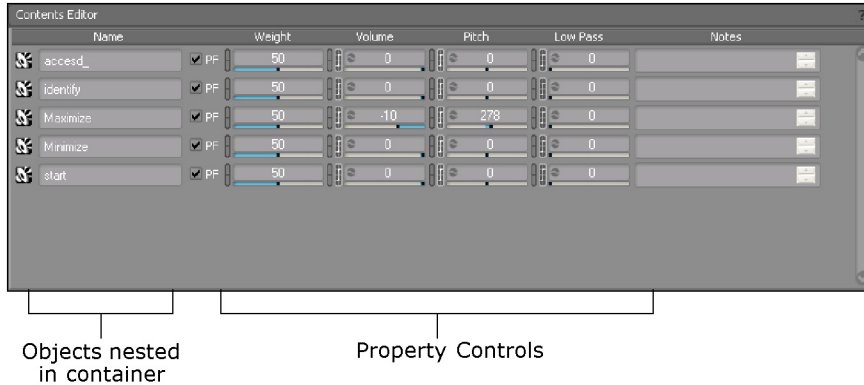
- **Takes** - Different versions of the same motion source that you can audition and test before choosing which one you will use. These motion sources can be created from audio files, plug-ins, or a combination of the two.
- **Motion Devices** - Different versions for the motion devices that you support in your project.

The controls and properties for each type of source in the Contents Editor will be different depending on the type of source that is selected.



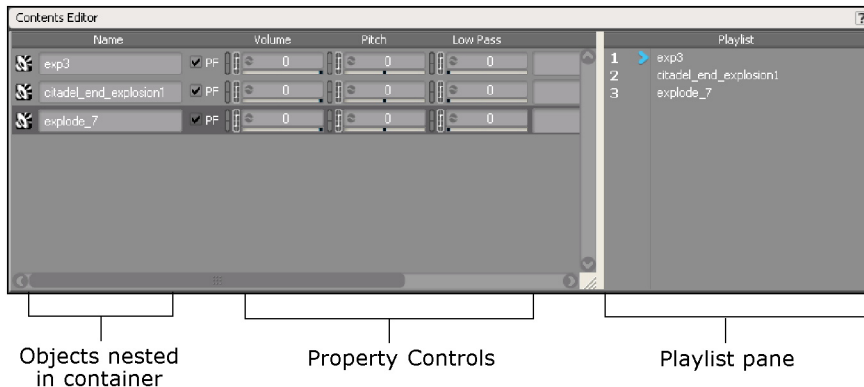
Random Container

When you load a random container into the Property Editor, its child objects are displayed in the Contents Editor where you can edit each object's properties.



Sequence Container

When you load a sequence container into the Property Editor, the child objects are displayed in the Contents Editor. You can edit each object's properties and create the playlist for the sequence container.



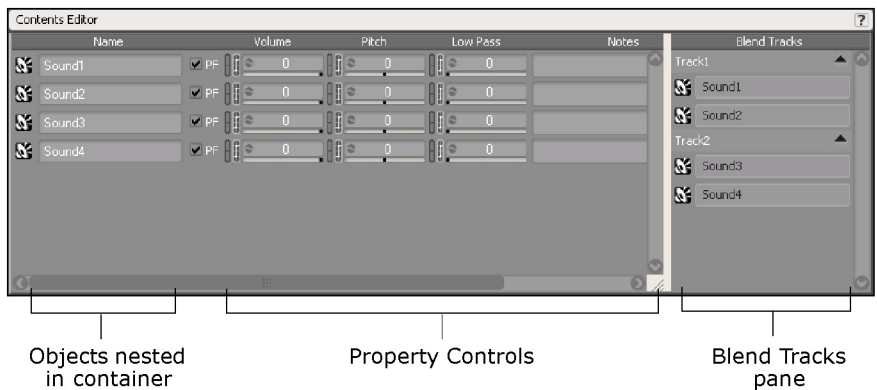
Switch Container

When you load a switch container into the Property Editor, the child objects are displayed in the Contents Editor. You can edit the properties for the child objects, assign objects to switches, and define the behavior for each object when the game calls a switch.



Blend Container

When you load a blend container into the Property Editor, the child objects are displayed in the Contents Editor. You can edit the properties for the child objects and assign them to the different blend tracks.



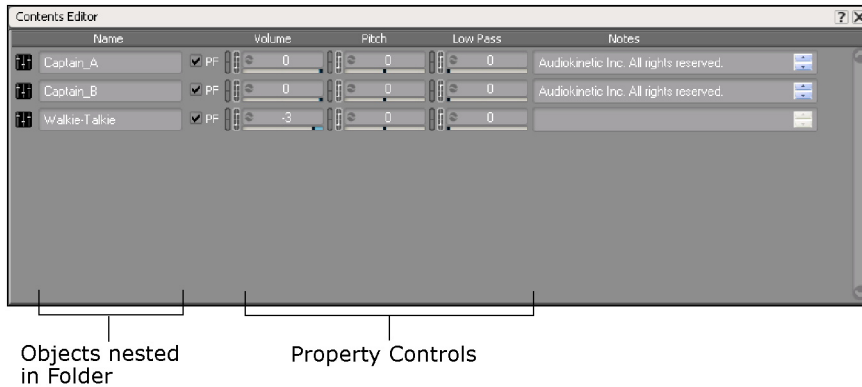
Actor-Mixer

When you load an Actor-Mixer into the Property Editor, the child objects are displayed in the Contents Editor. You can edit the properties for the child objects of the Actor-Mixer in this view.



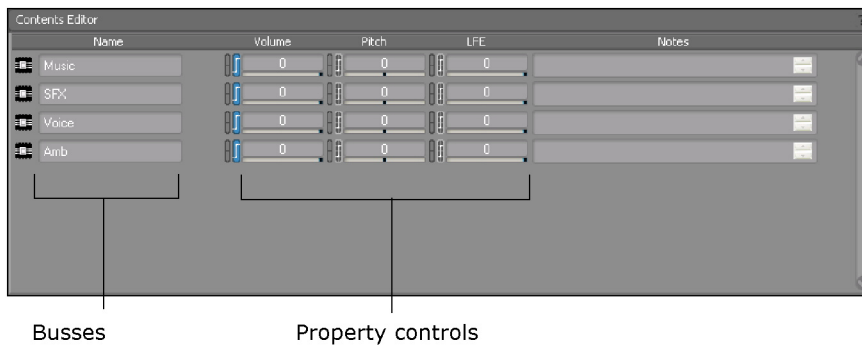
Folder

When you load a folder into the Property Editor, the child objects are displayed in the Contents Editor. You can edit the properties for the child objects of the folder in this view.



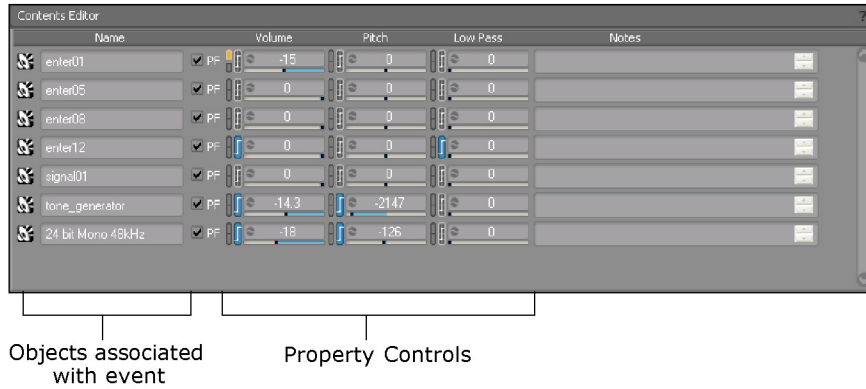
Bus

When you load the master bus or a parent bus into the Property Editor, its child busses are loaded into the Contents Editor. You can edit the child bus properties in this view.



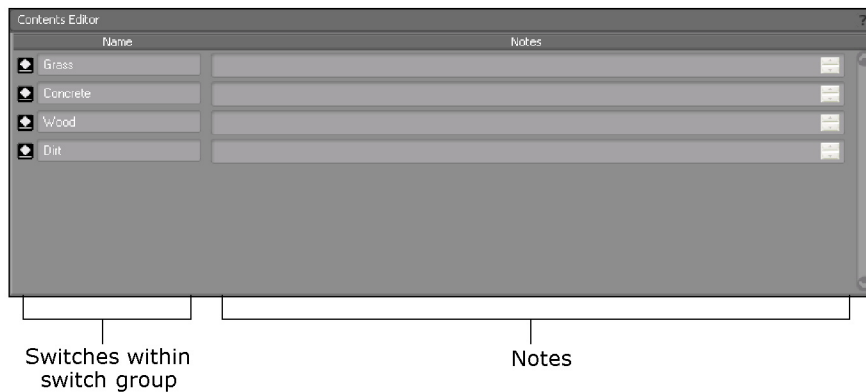
Event

When you load an event into the Event Editor, the objects associated with the event will be displayed in the Contents Editor. You can edit the properties for these objects.



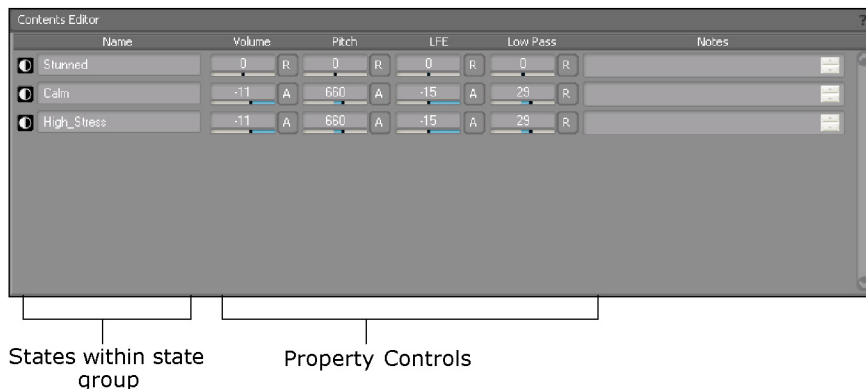
Switch Group

When you load a switch group into the Property Editor, the switches within the switch group will be displayed in the Contents Editor. You can edit the name and add notes for each of these switches.



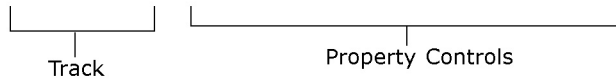
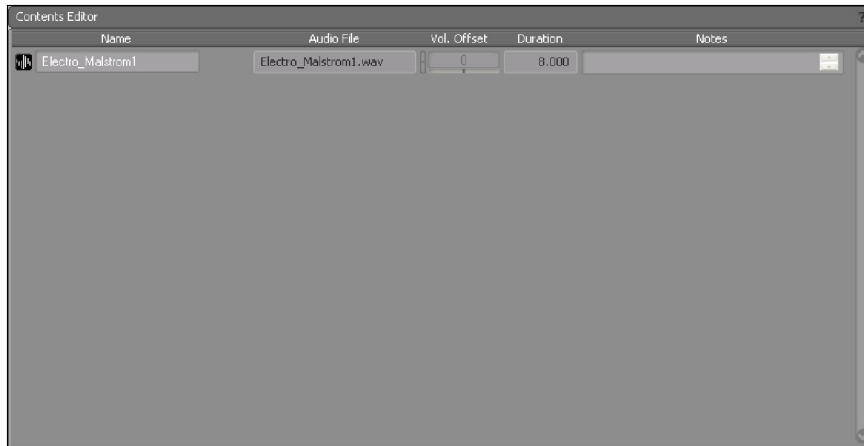
State Group

When you load a state group into the Property Editor, the states within the state group will be displayed in the Contents Editor. You can edit the properties for each of these states.



Music Track

When you load a music track into the Property Editor, it is displayed in the Contents Editor. Its properties are displayed, but only its notes can be edited here.



Music Segment

When you load a music segment into the Property Editor, its tracks are displayed in the Contents Editor where you can edit the object properties.



Music Playlist Container

When you load a music playlist container into the Property Editor, its child objects are displayed in the Contents Editor where you can edit the object properties.



Music Switch Container

When you load a music switch container into the Property Editor, its child objects are displayed in the Contents Editor where you can edit the object properties.










Working with the Contents Editor

The Contents Editor contains a series of fields, lists, options, and sliders that you can use to define properties and behaviors for your objects, states, and switches. If you require help using these tools, refer to the following sections:

- [Using Text Boxes](#)
- [Using Lists](#)
- [Using Sliders](#)

In the Contents Editor, you can play back individual objects, and re-order, copy, paste, and delete objects within the view. The options or properties that appear in the view will be different depending on the object you currently have loaded. However, in most views certain icons are displayed beside some of the property sliders to indicate that certain properties have been assigned to them. The different icons are described in the following table.

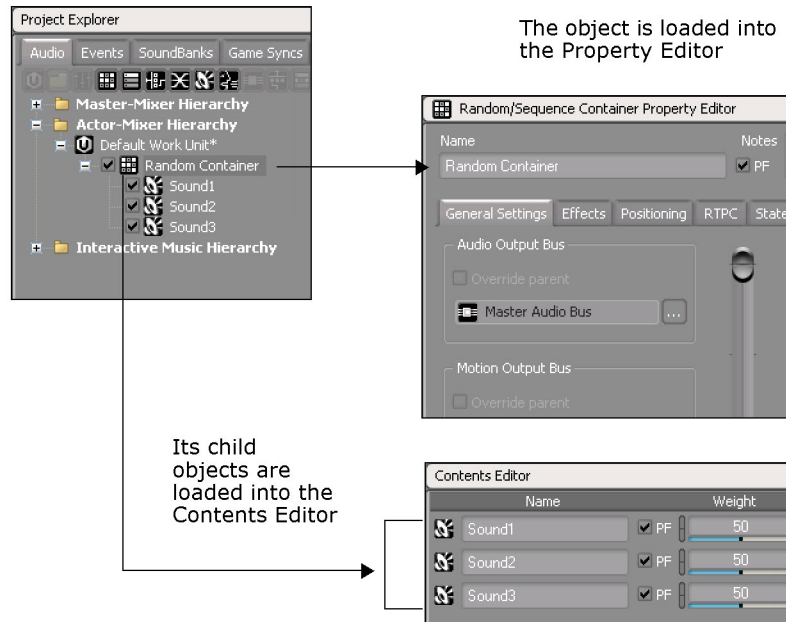
Icon	Name	Description
	Link	The property value is linked to the values of other active game platforms.
	Unlink	The property value is not linked to the values of the other active game platforms.
	Partial Unlink	The property value for the current platform is linked to the other platforms, but one or more corresponding values of other active platforms are unlinked.
	RTPC - Disabled	This property value is not tied to a game parameter value.
	RTPC - Enabled	A game parameter value is tied to this property value. This means, for example, that the speed of a car in game can be tied directly to the pitch property in Wwise. As the speed of the car increases in game, the pitch in Wwise will increase in real time.
	Randomizer - Enabled	A property value to which a Randomizer effect has been applied.
	Randomizer - Disabled	A property value to which no Randomizer effect has been applied.

Displaying Objects in the Contents Editor

When you load an object from the hierarchy into the **Property Editor**, its child objects will be displayed in the Contents Editor. For example, if you select a music switch container named "Action Sounds" in the **Audio** tab of the **Project Explorer**, and load it into the **Property Editor**, its child segments will be loaded into the **Contents Editor**.

Getting to Know the Contents Editor

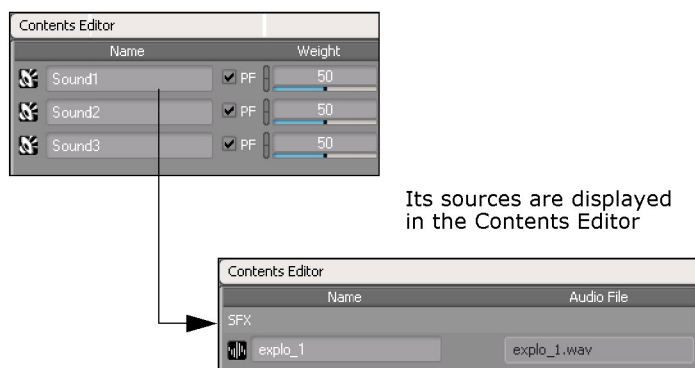
Double-click an object in the Audio tab of the Project Explorer



You can also move down the project hierarchy by double-clicking an object in the Contents Editor.

You can continue to move down the project tree right down to the source(s) with which an object is associated.

Double-click an object in the Contents Editor



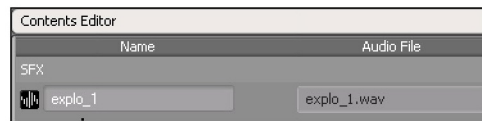
Note

To navigate back to the previous object, press **Backspace**. To navigate up to the current object's parent, press **Alt+Up Arrow**.

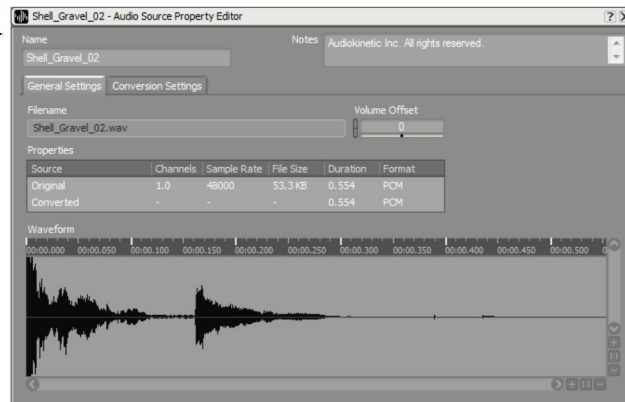
Working with Sources in the Contents Editor

Once you are at the source level in the **Contents Editor**, you can work with the settings for audio sources or source plug-ins. When you double-click an audio source, the **Conversion Settings** dialog box opens.

Double-click a source in the Contents Editor

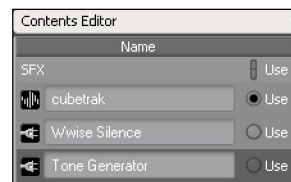


The Conversion Settings dialog box opens

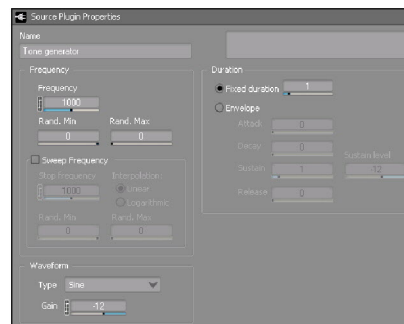


When you double-click a source plug-in, the plug-ins properties are displayed in the **Property Editor** where you can edit them.

Double-click a source in the Contents Editor



The Source Plug-ins properties are displayed in the Property Editor



Adding Objects to the Contents Editor

When you load an object in the **Property Editor**, its contents are simultaneously displayed in the **Contents Editor**. You can also add objects directly into the **Contents Editor** by dragging them from the **Project Explorer** into the **Contents**

Editor. Keep in mind that whenever you drag an object into the **Contents Editor**, you are moving that object out of its current location in the hierarchy to a new location under the parent of the other objects in the Contents Editor. If you do not want to move the object, you can Ctrl+drag the object into the **Contents Editor**. This will copy the object and add it to the other objects in the **Contents Editor**.

Sources can also be added to a sound or motion object by importing directly into the **Contents Editor** or by adding source plug-ins. If you want to convert an imported file, you can double-click the source to open the **Conversion Settings** dialog box.

For more information about importing and converting audio files, refer to the following sections.

- [Chapter 6, *Managing Media Files in Your Project*](#)
- [Converting Audio Files](#)



Note

Objects in the Contents Editor appear in the order that you have set them. Objects that have been added to the Contents Editor appear in the Audio tab of the Project Explorer under their parent in alphabetical order.

Re-ordering Objects within the Contents Editor

You can change the order for certain objects or sources in the Contents Editor as needed. This will not affect their order in the Project Explorer.

To change the order of objects:

1. In the Contents Editor, drag the object that you want to move to its new location.

A red indicator appears above the area where the object will be inserted.

Related Topics

- [Adding Objects to the Contents Editor](#)
- [Dragging Objects Between the Panes of the Contents Editor](#)
- [Expanding/Collapsing Lists](#)
- [Deleting Objects](#)
- [Auditioning Objects and Sources within the Contents Editor](#)

Dragging Objects Between the Panes of the Contents Editor

When you are working with switch, sequence, and blend containers, the Contents Editor is split into two separate panes. The first pane contains the child objects of the container and the second pane contains a list. You can drag objects from one pane to the other to create playlists or to assign objects to switches or blend tracks.

When you want to assign an object to a switch, you can drag that object from the Objects pane into the Assign Objects pane. You can also drop an object from the Audio tab of the Project Explorer onto a switch. For more information about working with objects and switches, refer to [Defining the Contents and Behavior of Switch Containers](#).



Note

Italics are used to denote that a switch is empty, and has no objects assigned to it.

When you want to create a playlist for a Sequence container, you can drag the object from the Objects pane into the Playlist pane. For more information about working with playlists and sequence containers, refer to [Creating a Playlist](#).

When you want to assign an object to a blend track, you can drag that object from the Objects pane into the Blend Tracks pane. You can also drop an object from the Audio tab of the Project Explorer onto a blend track. For more information about working with blend tracks, refer to [Adding and Removing Objects from Blend Tracks](#).

Expanding/Collapsing Lists

To make it easier to work with objects in the panes of the Contents Editor, you can expand or collapse lists. This feature is available for the following:

- **Switch containers** - To expand and collapse the Assigned Object lists.
- **Blend containers** - To expand and collapse the Blend Tracks lists.
- **Languages** - To expand and collapse the source list for each language.
- **Motion devices** - To expand and collapse the source list for each motion device.

To expand or collapse a list:

1. Do one of the following:

Click the arrow beside the list that you want to collapse.

The list is collapsed.

Click the arrow beside the list that you want to expand.

The list is expanded.

To expand or collapse all the language or motion device lists:

1. Right-click any language or motion device title bar.

The shortcut menu opens.

2. Do one of the following:

To expand every language or motion device list, click **Expand All**.

To collapse every language list or motion device, click **Collapse All**.

Related Topics

- [Re-ordering Objects within the Contents Editor](#)
- [Dragging Objects Between the Panes of the Contents Editor](#)
- [Adding Objects to the Contents Editor](#)
- [Deleting Objects](#)
- [Auditioning Objects and Sources within the Contents Editor](#)

Deleting Objects

If you no longer need an object or source, you can delete it in the Contents Editor. You may have imported several sources for testing purposes and after you decide which version you want, you can delete the rest. When you delete an object or a source in the Contents Editor, you are deleting the object or source from your project. This will not automatically delete the associated converted audio file from your project .cache folder. To delete the orphan file, you need to clear your audio cache. For more information about how to manage these orphan files, refer to [Clearing Your Cache](#).

To delete an object in the Contents Editor:

1. Select the icon for the object that you want to delete.
2. Do one of the following:

Press the **Delete** key.

Right-click the object, and from the shortcut menu, select **Delete**.

The selected object or source is removed from the Contents Editor and the Audio tab of the Project Explorer.



Note

Deleting an object from the Contents Editor deletes the object from the project. When you remove an item from the playlist or the switches list, you are not deleting the object. It is simply removed from the list.

Related Topics

- [Re-ordering Objects within the Contents Editor](#)
- [Dragging Objects Between the Panes of the Contents Editor](#)
- [Expanding/Collapsing Lists](#)
- [Adding Objects to the Contents Editor](#)
- [Auditioning Objects and Sources within the Contents Editor](#)

Auditioning Objects and Sources within the Contents Editor

You can audition each object and source in the Contents Editor using the Transport Control. This is very useful when you are trying to decide which source to use for a sound or motion FX object, or when you want to audition a playlist that you have created.

To audition objects in the Contents Editor:

1. Load an object into the Transport Control.
2. In the Transport Control, click the **Play** icon.

The selected object plays back.

3. Use the other controls in the Transport Control to pause or stop playback.

For more information about auditioning in the Transport Control, refer to [Chapter 40, *Getting to Know the Transport Control*](#).

To audition sources in the Contents Editor:

1. Load a sound, voice, motion FX, or music track object into the Transport Control.
2. If there is more than one source, select the Use option adjacent to the source that you want to play.
3. In the Transport Control, click the **Play** icon.

The selected source plays back.

4. Use the other controls in the Transport Control to pause or stop playback.

For more information about auditioning in the Transport Control, refer to [Chapter 40, *Getting to Know the Transport Control*](#).

Related Topics

- [Adding Objects to the Contents Editor](#)
- [Re-ordering Objects within the Contents Editor](#)
- [Dragging Objects Between the Panes of the Contents Editor](#)
- [Expanding/Collapsing Lists](#)
- [Deleting Objects](#)

Chapter 40. Getting to Know the Transport Control

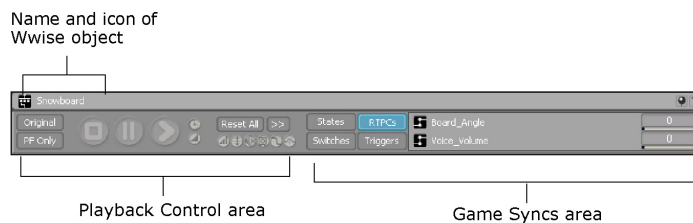
Overview	826
Setting Playback Properties	828
Pinning an Object in the Transport Control	831
Playing/Pausing/Stopping Content	832
Using Game Syncs During Playback	833

Overview

When you are editing sound, music, or motion properties you need to be able to audition your work. In Wwise, when you select a sound, music, or motion FX object, container, or event, it is automatically loaded into the Transport Control where you can audition it. The name of the object along with its associated icon are displayed in the title bar.

The Transport Control consists of two different areas:

- [Playback Control Area](#)
- [Game Syncs Area](#)

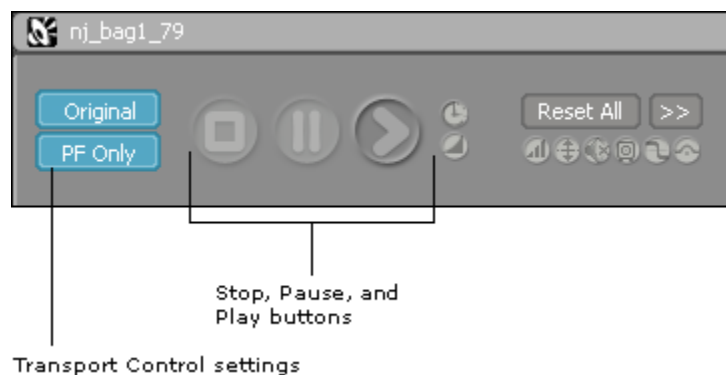


Note

Dialogue events are not loaded into the Transport Control; however, the objects assigned to a path can be loaded.

Playback Control Area

The Transport Control contains the traditional controls associated with the playback of audio, such as play, stop, and pause buttons. You can also determine how objects will be played back using the Transport Control settings. By selecting or deselecting these settings, you can specify whether the original or converted object is played and whether objects excluded from the current platform will be played or not.





Note

You cannot play back an actor-mixer or bus, so they are not loaded into the Transport Control when selected.

The Playback Control area also contains a series of indicators that change color when certain properties or behaviors have been previously applied to the object that is playing. The following table lists the property and action parameter indicators in the Transport Control.

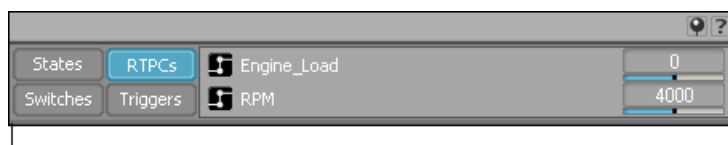
Icon	Name	Indicates
	Delay	A delay has been applied to an object in an event or a random or sequence container.
	Fade	A fade has been applied to an object in an event or a random or sequence container.
	Set Volume	A set volume action has been applied to an object in an event.
	Set Pitch	A set pitch action has been applied to an object in an event.
	Mute	A mute action has been applied to an object in an event.
	Set Low Pass Filter	A set Low Pass Filter action has been applied to an object in an event.
	Enable Bypass	An Enable Bypass action has been applied to an object in an event.

For more information on editing these properties for objects, refer to the following sections:

- [Playing All Objects Within the Container](#)
- [Setting Properties for an Event Action](#)

Game Syncs Area

In addition to the traditional playback controls, the Transport Control has a Game Syncs area that contains all the states, switches, RTPCs and Triggers associated with the currently selected object. You can use the Transport Control as a mini simulator to test out your sounds, music, and motion, and simulate changes in the game. During playback, you can change between states and switches, and audition the game parameters and their mapped values.



Game Syncs area

For more information on working with game syncs, refer to the following sections:

- [Chapter 16, Working with States](#)
- [Managing the Contents of a Switch/State](#)
- [Managing Game Parameters Used in RTPCs](#)
- [Working with Triggers](#)

Setting Playback Properties

By enabling various controls in the Transport Control, you can do any of the following:

- [Using Original Audio Files during Playback.](#)
- [Including/Excluding Audio and Motion Content for Playback.](#)
- [Resetting in the Transport Control](#)

Using Original Audio Files during Playback

When you convert your imported audio files, Wwise maintains an original version of the audio file that you can audition whenever you want. This original version has not been converted for platforms. By default, the Transport Control plays the original sounds; however, you can choose to play the converted version.

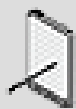
To play back original sounds:

1. In the Transport Control, click **Original**.

The button becomes blue.

2. Click the **Play** icon.

The original pre-converted sounds will play for the object in the Transport Control.



Note

To play back the converted sounds, click the **Original** button to deselect it.

Related Topics

- [Including/Excluding Audio and Motion Content for Playback](#)
- [Resetting in the Transport Control](#)

Including/Excluding Audio and Motion Content for Playback

When you are creating the audio, music, and motion structures, you can decide to include or exclude certain objects from one or more platforms. For more information about working with platforms, refer to [Excluding Project Elements from a Platform](#). When you are playing back sounds, music, or motion FX, you may choose to play only the content that is in the current platform, or play all sounds, music, or motion FX loaded into the Transport Control.

To play back platform-specific content:

1. From the Platform Selector list on the toolbar, select the platform for the objects that you want to audition.
2. In the Transport Control, click **PF Only**.

The PF Only button turns blue and only the objects and events in the current platform will be played in the Transport Control.



Note

To play back all objects and events, click **PF Only** again to disable this option.

Related Topics

- [Including/Excluding Audio and Motion Content for Playback](#)
- [Using Original Audio Files during Playback](#)

Resetting in the Transport Control

When you play back objects in the Transport Control, you have access to a range of properties, behaviors, and game syncs for the objects and this can help you simulate the in-game experience. When you are connected to the game, some game syncs, effects, and events may affect the previously defined properties for objects as well. The property indicators in the Transport Control provide you with feedback about which behaviors or actions are still in effect during playback. To return objects to their previous settings, you can use the reset function.



Note

For each event playback instance, the event action property value is added to the object's properties. When you are working with

these event actions, you should reset these action properties before replaying the object to clear these cumulative properties and restore the default. To know more about how event actions affect object properties, refer to [Working with Events](#).

To reset specific transport control properties:

1. In the Transport Control, click the Selector icon.

The Reset menu is displayed.

2. From the Reset menu, select one of the following:

Reset All to reset all objects to their original settings.

Reset All Random and Sequence Containers to resume playing a sequence container from the beginning of the playlist or to reset random playback properties for random containers.

Reset All Game Parameters to return all game parameters to the original settings.

Reset All Set Mute to clear all mute actions that have been triggered for the objects.

Reset All Set Pitch to clear all pitch actions that have been triggered for the objects.

Reset All Set Volume to clear all volume actions that have been triggered for the objects.

Reset All Set Low Pass Filter to clear all Low Pass Filter actions that have been triggered for the objects.

Reset All Bypass Effect to clear all Bypass actions that have been triggered for the objects.

Reset All States to return to the default state.

Reset All Switches to return to the default switch specified for the Switch Container.

Reset All Music Tracks Force Usage to clear any music tracks from being forced to play in the Transport Control.

Reset Position to reset the position of the listener within the Attenuation Preview control to its default position.



Tip

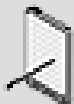
To reset all objects to their default settings without displaying the Reset menu, click **Reset All**.

Related Topics

- [Including/Excluding Audio and Motion Content for Playback](#)
- [Using Original Audio Files during Playback](#)

Pinning an Object in the Transport Control

The Transport Control automatically loads whatever object is currently shown in the Property Editor. However, when you select any object or event in the Project Explorer, by default it will replace whatever is in the Transport Control. If you want to keep an object loaded in the Transport Control regardless of whatever else you select, you can pin that object, and prevent other objects from being loaded.



Note

When you are mixing in real-time, you need to load the object into the Transport Control so that your changes to the relative properties for that object take effect. Ensure that no objects are pinned when you are mixing in real-time.

Using Pinning - Example

Let's say that you are auditioning a random container and you are curious to see how a reverb effect will sound on a particular object in the container. You could pin the object in the Transport Control. Then you could apply the reverb to the parent object in the Property Editor and audition the effect for that object alone. If you did not pin the object, all the container objects would play back in random mode and the object you wanted to audition might not play back. By pinning the object, you can modify the parent settings and then audition an object again as often as you need to.

To pin an object loaded in the Transport Control:

1. Load an object into the Transport Control.
2. In the Transport Control, click **Pin**.

The Pin icon will become red and the object or event will remain in the Transport Control even when you load other objects or events into the Property Editor, or select another object in the Project Explorer.



Tip

Press **Alt+ Click** to load and pin an object in the Transport Control. To unpin an object, press **Ctrl + Alt + p**.

Playing/Pausing/Stopping Content

If you want to audition an object, you can use the standard playback controls in the Transport Control.



Shortcut keys have also been mapped to the main control buttons in the Transport Control view to make it easy for you to start, pause, and stop playback while editing property values in real time. For a complete list of keyboard shortcuts, refer to [Appendix B, Shortcuts](#).

To play back sounds in the Transport Control:

1. Load an object into the Transport Control.
2. Do any of the following:

Click the **Play** icon.

The associated content will play until finished.

To pause playback, click the **Pause** icon.

The Pause icon turns yellow. To resume play, click **Pause** again.

To stop playback, click the **Stop** icon.

Related Topics

- [Using Original Audio Files during Playback](#)
- [Including/Excluding Audio and Motion Content for Playback](#)
- [Pinning an Object in the Transport Control](#)
- [Using Game Syncs During Playback](#)

Using Game Syncs During Playback

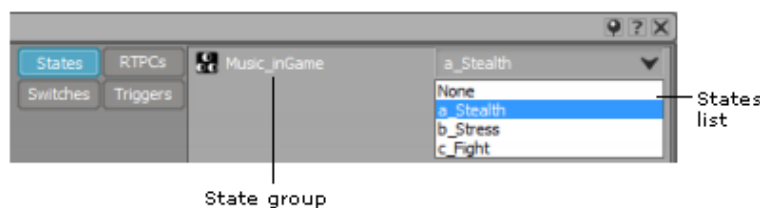
In the Transport Control you have access to the game syncs that you have created for your project. To audition the states, switches, and RTPCs as they are applied to your sounds, music, and motion FX, you can do any of the following:

- [Enabling States During Playback](#)
- [Assigning Switches During Playback](#)
- [Changing Game Parameter Values During Playback](#)
- [Calling Triggers During Playback](#)

Enabling States During Playback

When an object is loaded into the Transport Control, you can select from the list of state groups and states to which the object is subscribed to simulate states and state changes that will occur in game during playback. This means that while you are playing back objects, you can audition the state properties and also switch between states to audition the state changes. To learn more about creating states and the state properties and transitions that you will be auditioning in the Transport Control, as well as assigning objects to states, refer to the following sections:

- [Working with States](#)
- [Assigning States to Objects and Busses](#)



To enable a state during playback:

1. Load an object into the Transport Control.
2. In the states list, select the state that you want to apply.

The state will be applied to all sounds in the Transport Control that subscribe to the state during playback.

3. Click the **Play** icon.

The state that you selected will be applied during playback. While the object is playing, you can continue to change states to simulate the game.



Note

To return to the default state, click **Reset > Reset All States**.

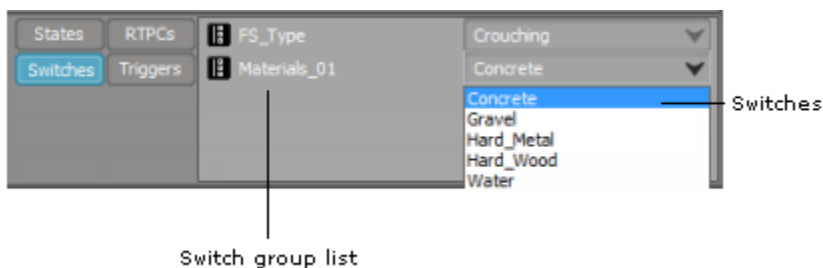
Related Topics

- [Assigning Switches During Playback](#)
- [Changing Game Parameter Values During Playback](#)

Assigning Switches During Playback

When an object is loaded into the Transport Control, you can select from the list of switch groups and switches to which the object has been assigned to simulate switch changes that will occur in game during playback. This means that while you are playing back objects, you can change switches and audition the changes. To learn more about creating switches and how switches are used, refer to the following sections:

- [Working with Switches](#)
- [Defining the Type of Switch Container](#)



To assign a switch during playback:

1. Load an object into the Transport Control.
2. From the switches list, select the switch that you want to apply.

The switch containers that have subscribed to the selected switch group will play the objects that correspond with the switch that you have chosen.

3. Click the **Play** icon.

The switch that you selected will be applied during playback. While the object is playing, you can continue to change switches to simulate the game.



Note

To return to the default switch for the Switch Container specified in the Property Editor, click **Reset > Reset All Switches**.

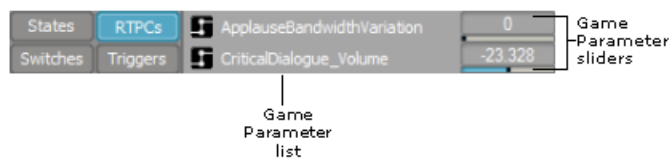
Related Topics

- [Enabling States During Playback](#)
- [Changing Game Parameter Values During Playback](#)

Changing Game Parameter Values During Playback

When an object is loaded into the Transport Control, the associated RTPCs are displayed in the Games Syncs area. A slider is provided so that you can change the game parameters while you are playing back your object. Since you have already mapped these values to Wwise property values, when you change the game parameter values, you automatically change the object property values. This simulates what happens in game when the game parameters change and you can verify how effectively your property mappings will work in game. To learn more about creating game parameters and mapping property values to them, refer to the following sections:

- [Managing Game Parameters Used in RTPCs](#)
- [Assigning Wwise Properties to Game Parameters](#)



You can audition these property changes during playback in your simulation.

To modify game parameter values during playback:

1. Load an object into the Transport Control.
2. In the Game Syncs area, click **RTPCs**.

The game parameters that have been mapped to the object will be displayed.

3. Click the **Play** icon.

While the object is playing, you can use the RTPC slider to change the game parameter values to see how your sound reacts to the changes.



Note

To return the game parameters to their default settings, click **Reset > Reset All Game Parameters**.



Note

To bypass game parameter interpolation, hold Ctrl while using the RTPC slider in **Transport Control**, or the game parameter cursor in the RTPC tab of the **Property Editor**.

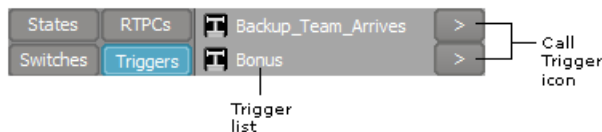
Related Topics

- [Assigning Switches During Playback](#)
- [Enabling States During Playback](#)

Calling Triggers During Playback

In addition to auditioning this object, you can also select from the list of triggers to audition stingers. A stinger is a brief musical phrase that is superimposed and mixed over the currently playing music. In this way you can simulate what is happening at key points in the game when a trigger calls a stinger to play over the current music. To learn more about creating triggers and creating the stingers for them that you will be auditioning in the Transport Control, refer to the following sections:

- [Working with Triggers](#)
- [Chapter 28, Using Stingers](#)



To call a trigger during playback:

1. Load a music object into the Transport Control.
2. Click the **Play** icon.

The music object loaded into the Transport Control will play back.

3. In the Game Syncs area, click the **Triggers** button to display the trigger list.



4. Click the **Call Trigger** icon.

The corresponding stinger will play over the currently playing music object. You can select other triggers and play back the corresponding stingers to simulate the music in the game.

Related Topics

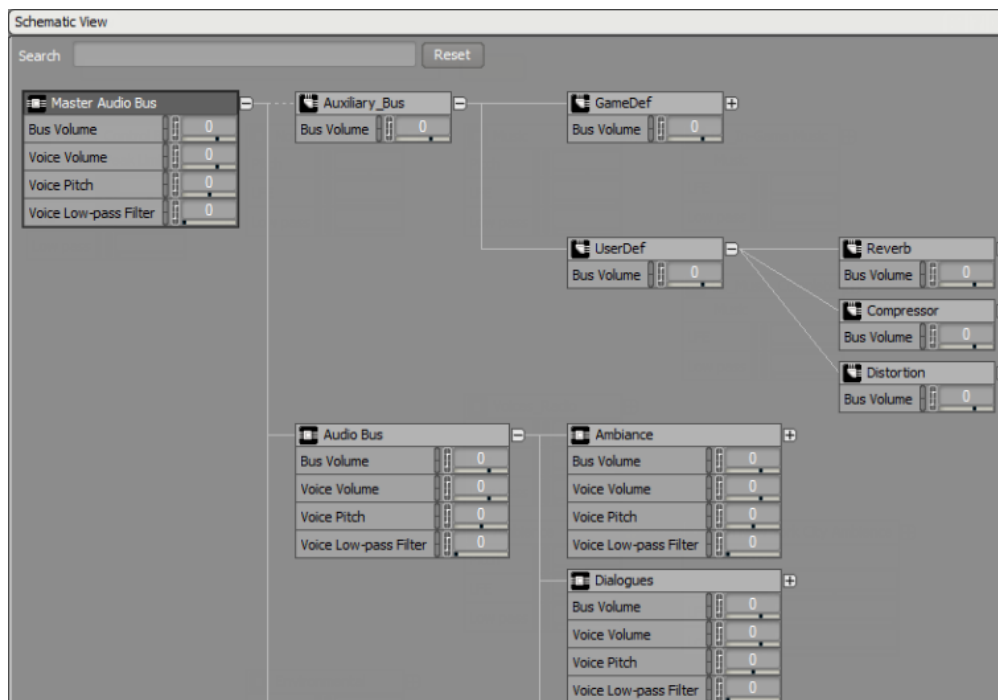
- [Playing/Pausing/Stopping Content](#)
- [Working with Triggers](#)
- [Chapter 28, *Using Stingers*](#)

Chapter 41. Getting to Know the Schematic View

Overview	839
Customizing the Schematic View	839
Working with the Schematic View	841

Overview

The Schematic view displays a graphical representation of the structure of your Wwise project. You can use the Schematic view to get an overview of a project, locate project objects, or analyze project structure one object at a time. The Schematic view includes icons representing each project object, the object names, and lines and nodes representing their relationships. You can customize how the Schematic view displays project object details as well.



Customizing the Schematic View






The Schematic view contains a visual representation of project objects, as well as specific tools to help customize this view. It also features a search function to quickly locate objects.

To customize the Schematic view, you can:

- [Identifying Project Objects and Connectors](#)
- [Setting the Schematic View Display Options](#)

Identifying Project Objects and Connectors

The Schematic view is made up of icons representing project objects and connectors representing their relationship to one another. The following table describes the connectors between project objects. For information on the icons representing project objects, refer to [Understanding the Visual Elements in Wwise](#).

Icon	Name	Description
	Solid line	These lines connect parent and child objects.
	Dashed line	These lines connect busses to child objects to demonstrate routing.
	Plus sign (white)	Click to expand the schema and show all children of the object.
	Plus sign (yellow)	Click to show all children of the object, when not all children are currently displayed.
	Minus sign	Click to collapse all children of the object.

Setting the Schematic View Display Options

You can customize the information shown about each project object in the schema by using the Schematic View Settings.

To specify the information to be displayed in the Schematic view:

1. Click the options icon in the upper right corner of the Schematic view.

The Schematic View Settings dialog box opens.

2. Select one or more of the following information types:

Icon Strip to display the row of icons representing object properties. If the object property has been changed, the icon is displayed in white and you can mouse over it for more information. If the property still has its default value, it is grayed out.

Mute/Solo to display the mute and solo buttons for each object.

Bus to display the bus through which the object is routed.

Conversion Settings to display the conversion settings ShareSet that is being used by the object.

Effect to display any effects that have been applied to the object.

Positioning Type to display the type of positioning (2D, 3D user-defined, 3D game-defined) applied to the object.

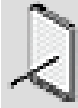
Game Parameters to display the game parameters affecting the object (through RTPCs).

State Group to display the state group to which the object subscribes.

Advanced Settings to display any changes made to the advanced settings of an object (such as playback limit or volume threshold).

Volume to display the volume assigned to the object.

Pitch to display the pitch assigned to the object.



Note

Pitch is not displayed for music objects, because it is not available for them.

Low Pass to display the low pass filter assigned to the object.

3. Click OK.

The Schematic View displays the information you selected for each project object.

Related Topics

- [Identifying Project Objects and Connectors](#)
- [Searching for Project Objects](#)
- [Showing a Project Object](#)
- [Editing Project Objects](#)

Working with the Schematic View

The Schematic view gives you multiple options for finding, examining, and working with the project objects displayed within it.

When working with the Schematic view, you can:

- [Searching for Project Objects](#)
- [Showing a Project Object](#)
- [Editing Project Objects](#)

Many of the commands in the following sections have been mapped to keyboard shortcuts. For a complete list of shortcuts, refer to [Appendix B, Shortcuts](#).

Searching for Project Objects

You can find a project object quickly in the Schematic view by using the Search function.

To search for a project object in the Schematic view:

1. In the **Search** field, type the name of the project object you want to find.

- Wwise highlights project objects matching the name you have entered.
2. Click **Reset** to erase the Search field and reset the Schematic view.



Note

The search is not case-sensitive, and will find objects as you type in your search term. The search engine looks at the beginning of words, so typing “gun” will find “gun”, “big_gun”, and “gunner”, but not “shotgun”.

Related Topics

- [Setting the Schematic View Display Options](#)
- [Identifying Project Objects and Connectors](#)
- [Showing a Project Object](#)
- [Editing Project Objects](#)

Showing a Project Object

You can simplify the project schema presented in the Schematic view by specifying one project object to be highlighted.

To highlight a project object in the Schematic view:

1. In the Schematic view, right-click a project object.

The shortcut menu is displayed.

2. From the shortcut menu, select **Show in Schematic View**.

The bus routing for the selected object is displayed. The object is highlighted, and unrelated objects are hidden.

3. To reset the Schematic view, click **Clear**.



Tip

You can also drag an object from the Project Explorer to show it in the Schematic view. You can also right-click in the Property or Contents editors and select **Show in Schematic View**.

Related Topics

- [Identifying Project Objects and Connectors](#)

- [Setting the Schematic View Display Options](#)
- [Searching for Project Objects](#)
- [Editing Project Objects](#)

Editing Project Objects

You can edit the project objects displayed in the Schematic view in the following two ways:

- [Editing Project Objects Directly](#)
- [Editing Project Objects in the Property Editor](#)

Editing Project Objects Directly

You can use the controls displayed under each project object in the Schematic view to edit the object. These controls work identically to those in the Property Editor. For more information on using these controls, refer to [Wwise Interface Basics](#).



Use the Schematic View Settings to display these controls. For more information on these settings, refer to [Setting the Schematic View Display Options](#).

Editing Project Objects in the Property Editor

If you prefer to edit object properties directly, you can quickly open the Property Editor for a project object from the Schematic view.

To open the Property Editor for a project object:

1. In the Schematic view, right-click a project object.

The shortcut menu is displayed.

2. From the shortcut menu, select **Edit**.

The Property Editor for that project object opens.



Note

You can leave the Schematic view open while you work in this Property Editor. The changes you make to a project object in the Property Editor are reflected immediately in the Schematic view.

Related Topics

- [Identifying Project Objects and Connectors](#)
- [Setting the Schematic View Display Options](#)
- [Searching for Project Objects](#)
- [Showing a Project Object](#)

Chapter 42. Getting to Know the Graph View

Overview	846
Changing the Display of the Graph View	847
Working with Control Points in the Graph View	852
Working with Curves in the Graph View	855

Overview

Several views in Wwise include a graph. You can use these graphs to map the relationship between two variables, plot spatial positioning, and monitor performance.

There are several different graphs in Wwise. Although they generally support many of the same functions, they do have their differences. The different graphs are located in the following views:

- Project Settings (Obstruction/Occlusion tab)
- Property Editor (RTPC tab)
- Blend Tracks Editor
- Attenuation Editor
- Position Editor
- Switch Group Property Editor
- Music Fade Editor
- Performance Monitor
- Game Sync Monitor
- Wwise Motion Generator Property Editor
- SoundSeed Air Source Plug-in Editor

The following table describes the unique purpose of each graph.

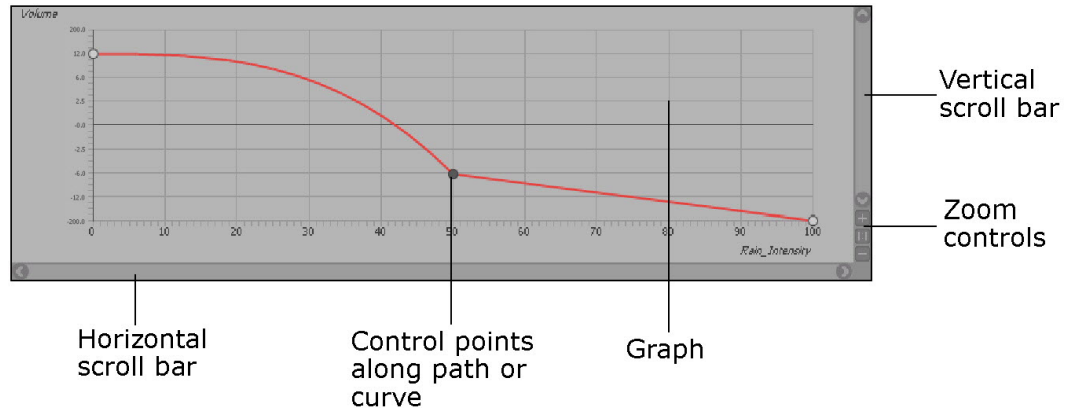
Use this graph	To
Project Settings (Obstruction/Occlusion tab)	Define the relationship between volume and LPF and the obstruction and occlusion factors for sound objects.
Property Editor (RTPC tab)	Define the relationship between game parameter values and the property values in Wwise.
Blend Tracks Editor	To group objects into blend tracks so that you can apply one or more RTPC curves to them, crossfade between them, or both.
Attenuation Editor	Define the relationship between certain properties and the distance the listener is from the emitting source to create realistic attenuation or fall-off of a sound.
Position Editor	Create a path within the 3D environment to localize the sound within the surround sound speaker environment.
Switch Group Property Editor	Map the different switches within a switch group to a specific game parameter.
Music Fade Editor	Define the fade in/out curves between two pieces of music in a music transition.
Performance Monitor	View information related to the performance of the sound engine as game elements are triggered by the game, game simulator, or Soundcaster. The information in the Performance Monitor is read-only and is for diagnostic purposes only. As a result, you cannot add and remove control points, or manipulate any of the curves.
Game Sync Monitor	Track the values of RTPCs as they are applied to the game objects you are watching.

Use this graph	To
	The information in the Game Sync Monitor is read-only and is for diagnostic purposes only. As a result, you cannot add and remove control points, or manipulate any of the curves.
Wwise Motion Generator Property Editor	Create a series of curves that define the intensity of the motion for the different motors available for each platform's controller.
SoundSeed Air - Woosh Source Plugin Editor (Object Path Graph)	Creates a path that defines the object's trajectory within a Woosh scene as well as to define the attenuation of the sound as it moves farther away.
SoundSeed Air (Wind/Woosh) Source Plugin Editor	Create a series of automation curves that define the properties of the generated wind or woosh sound.

When creating sound paths and monitoring performance, or working with segments, a time frame also needs to be defined. In these cases, the graph view is associated with a timeline. For more information about working with timelines in Wwise, refer to [Overview](#).

Describing Common Elements of the Graph

Although each of the graph views in Wwise are a little different, they do have some common elements, such as zoom controls, scroll bars, and control points that form a curve or path. The following illustration shows each of the common elements in the graph view.



Changing the Display of the Graph View

You can change the graph view display to help position control points more accurately or to examine specific areas more closely. You can use the following tools to change the graph view display:

- [Zooming and Panning the Graph View](#)
- [Defining the Scaling Method of the Graph View](#)
- [Showing/Hiding Grid Lines](#)
- [Showing/Hiding Cursors](#)

Zooming and Panning the Graph View

You can use the pan and zoom tools to position specific points along a path or curve, or to examine a specific area of information more closely. You can zoom in and out, pan up, down, left, and right.

All the pan and zoom controls have associated keyboard shortcuts.

To	Use this shortcut
Zoom in	Z+marquee selection
Zoom in – centered at current mouse position	Ctrl+mouse wheel up
Zoom out – centered at current mouse position	Ctrl+mouse wheel down
Reset zoom (when zoomed in)	Z+click
Pan view up (when zoomed in)	Mouse wheel up
Pan view down (when zoomed in)	Mouse wheel down
Pan view left (when zoomed in)	Shift+mouse wheel up
Pan view right (when zoomed in)	Shift+mouse wheel down
Pan free hand (when zoomed in)	X+drag

For a complete list of shortcuts, refer to [Appendix B, Shortcuts](#).

To zoom in the graph view:

1. From the graph view toolbar, click the **Zoom In** icon.

The graph zooms in towards the center of the graph view.

To zoom out the graph view:

1. From the graph view toolbar, click the **Zoom Out** icon.

The graph zooms out from the center of the graph view.

To reset the pan and zoom in the graph view:

1. From the graph view toolbar, click the **Reset** icon.

The graph view is reset to show all existing points.

To pan the graph view:

1. Zoom in the graph view.
2. Do one of the following:

To pan up, move the mouse wheel up.

To pan down, move the mouse wheel down.

To pan right, press **Shift** and move the mouse wheel down.

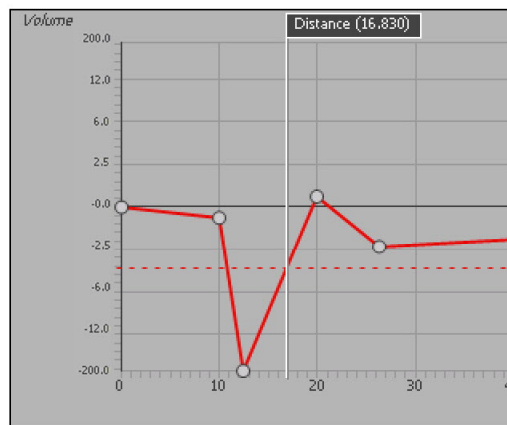
To pan left, press **Shift** and move the mouse wheel up.

To pan freehand, press **x** and drag the mouse in the graph view.

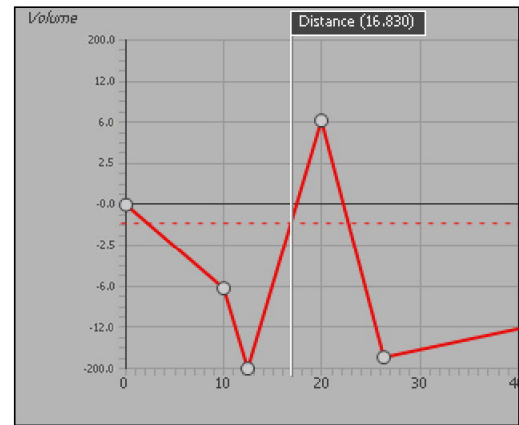
Defining the Scaling Method of the Graph View

When viewing curves that are measured in decibels, you have the option to display these curves in a strictly linear manner or according to a more normal (logarithmic) falloff of sounds in decibels. In linear scaling mode, there is an equal distance between intervals on the Y axis. In dB scaling mode, the Y axis emulates how the human ear interprets sounds at different decibels. This translates into wider gaps between intervals closer to zero and smaller distances for lower decibel values.

The following illustration shows the difference between showing the same volume curve in dB scaling and linear scaling.



Linear Scaling

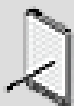


dB Scaling

Since dB scaling affects the distribution of decibel units along the Y axis, the scaling method will also affect how control points move within the graph view. For example, a change of -5 dB in dB scaling will require more movement when a point is at 0 dB than when it is at -80 dB. In linear scaling, however, a change of -5 dB will always result in the same movement along the Y axis.

In most cases, dB scaling will result in a more accurate representation of how sounds will be heard at a particular game parameter value. However, in cases where you want to create a direct relationship between a game parameter and a Wwise property, you will want to use Linear scaling. For example, let's say you want to map the volume of the Voices bus to a volume slider in the game that allows the game player to increase or decrease the volume of voices. In this case,

you will want a direct mapping of the volume of the sound with the volume in the game. Since the volume along the X axis is linear, the Y axis needs to be linear as well.



Note

The scaling method affects the units along the Y axis, so if two or more curves are displayed at the same time with different scaling methods, the units on the Y axis will not be displayed.

To define the scaling method of the graph view:

1. In the graph view, right-click a curve.

A shortcut menu is displayed.

2. Select one of the following scaling methods:

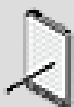
dB scaling to display the curves according to the normal (logarithmic) falloff of sounds in decibels as interpreted by the human ear.

Linear scaling to display the curves in a strictly linear manner.

The graph view scales the curves according the method you selected.

Showing/Hiding Grid Lines

Some of the graph views contain grid lines to help you position control points more accurately. You can show or hide these grid lines without affecting the points on the graph.



Note

The grid lines are displayed by default.

To hide the grid lines in the graph view:

1. Right-click in the graph view.

A shortcut menu is displayed.

2. Select one of the following:

Display Grid (Vertical) to remove the vertical grid lines.

Display Grid (Horizontal) to remove the horizontal grid lines.

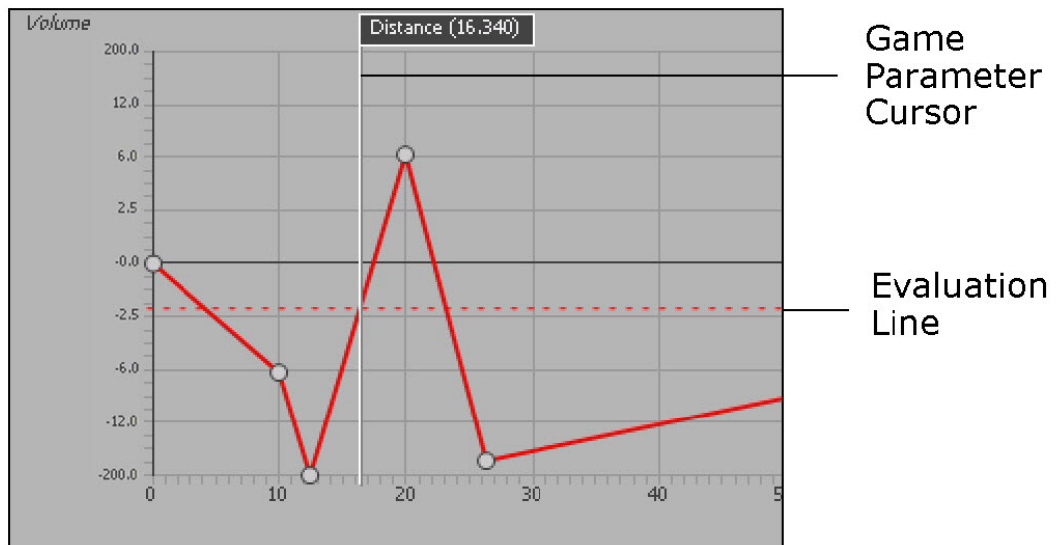
3. Repeat steps 1-2 to remove the remaining grid lines.

To show the grid lines in the graph view:

1. Right-click in the graph view.
A shortcut menu is displayed.
2. Select one of the following:
Display Grid (Vertical) to add the vertical grid lines.
Display Grid (Horizontal) to add the horizontal grid lines.
3. Repeat steps 1-2 to add the remaining grid lines.

Showing/Hiding Cursors

The RTPC graph views contain game parameter cursors and evaluation lines for each of the curves displayed in the graph view. The Game Parameter cursor has a tag at the top with the name of the corresponding game parameter and its current value along the curve. You can drag this cursor back and forth during playback to simulate changes in game parameter values. The evaluation lines are there to help you determine the property value that corresponds to a particular game parameter value. You can show or hide these cursors without affecting the points on the graph.



Note

The cursors and evaluation lines are displayed by default. In cases where several curves are displayed, the evaluation lines will only be displayed if the curves use the same scaling and the Y axes use the same unit.

To hide the cursors in the graph view:

1. Right-click in the graph view.

A shortcut menu appears.

2. Select **Show cursors**.

The cursors are removed from the graph view.

To show the cursors in the graph view:

1. Right-click in the graph view.

A shortcut menu appears.

2. Select **Show cursors**.

The cursors are displayed in the graph view.

Working with Control Points in the Graph View

Control points in the graph view define the shape of the curve or path, or create specific relationships between two variables. You can add, move, and delete a point along the curve at any time. You can also select several points at the same time to manipulate or delete a complete section of a curve or several curves.

All the information you need to know about control points is described in the following sections:

- [Adding Control Points](#)
- [Selecting Control Points](#)
- [Moving Control Points](#)
- [Deleting Control Points](#)

Adding Control Points

You can add points anywhere along the curve to define its shape or to create relationships between different variables. Since you are defining different types of information in each of the graph views, there are slight differences with the way you add control points. The following table shows you how to add points in the different graph views.

Graph	To add a point	To insert a point between two points
Project Settings (Obstruction/Occlusion tab)	NA	Double-click between two points on the curve.
RTPC/Blend Tracks Editor	NA	Double-click between two points on the curve.
Position Editor	Double-click	Ctrl+double-click between two points on the curve.
Attenuation Editor	NA	Double-click between two points on the curve.

Graph	To add a point	To insert a point between two points
Switch Group Property Editor	NA	Double-click between two points on the curve.
Music Fade Editor	N/A	N/A
Performance Monitor	N/A	N/A
Game Sync Monitor	N/A	N/A
Wwise Motion Generator Source Plugin Editor	NA	Double-click between two points on the curve.
SoundSeed Air - Woosh Source Plugin Editor (Object Path Graph)	Double-click	Ctrl+double-click between two points on the curve.
SoundSeed Air (Wind/Woosh) Source Plugin Editor	NA	Double-click between two points on the curve.

Selecting Control Points

Before you can move or delete a control point in the graph view, you must select it first. You can select one, several, or all control points at a time. If several curves are displayed in the graph view at the same time, you can select several points on different curves to move or delete them.

To select a control point in the graph view:

1. In the graph view, click a control point to select it.

The selected control point turns black.

To select several control points in the graph view:

1. In the graph view, do one of the following:

Drag a marquee selection over the points you want to select.

Ctrl+click the points you want to select.

The selected points turn black.

To select all control points in the graph view:

1. Click in the graph view to make it active.
2. Press **Ctrl+A**.

All points in the graph view become selected and turn black.

Moving Control Points

You can manipulate the shape of a curve or curves by moving one or more control points anywhere within the graph view. When the graph view is completely zoomed out, however, you are constrained by the outside borders of the graph view.

To move control points in the graph view:

1. In the graph view, select one or more control points.

The selected control points turn black.

2. Drag the point(s) anywhere within the graph view boundaries.



Note

You can also move points by using the arrows keys or by typing a value directly into the X and Y Coordinates fields. If more than one point is selected, the values you type in the X and Y fields will offset the points from their original location.

Deleting Control Points

If you want to remove points from your curves or path, you can delete them. Certain points, however, cannot be deleted. The following table describes the points that can be deleted in each of the graph views.

Graph	Points that can be deleted
Project Settings (Obstruction/Occlusion tab)	All points except the first and last one.
RTPC/Blend Tracks Editor	All points except the first and last one.
Attenuation Editor	All points except the first and last one.
Position Editor	All points except the first one.
Switch Group Property Editor	All points except the first and last one.
Music Fade Editor	N/A - there are no control points in the Music Fade Editor.
Performance Monitor	N/A - there are no control points in the Performance Monitor.
Game Sync Monitor	N/A - there are no control points in the Game Sync Monitor.
Wwise Motion Generator Source Plugin Editor	All points except the first and last one.
SoundSeed Air - Woosh Source Plugin Editor (Object Path Graph)	All points except the first one.
SoundSeed Air (Wind/Woosh) Source Plugin Editor	All points except the first and last one.

To delete control points from the curve:

1. Select one or more points along the curve(s) or path.

The selected control points turn black.

2. Press the **Delete** key.

The control point(s) are removed from the curve.

Working with Curves in the Graph View

Since you can be working on many curves at the same time within the same graph view, it is important that you learn to do the following basic tasks:

- [Displaying Curves in the Graph View](#)
- [To pin a curve to the graph view:](#)
- [Specifying the Shape of the Curve Between Control Points](#)



Note

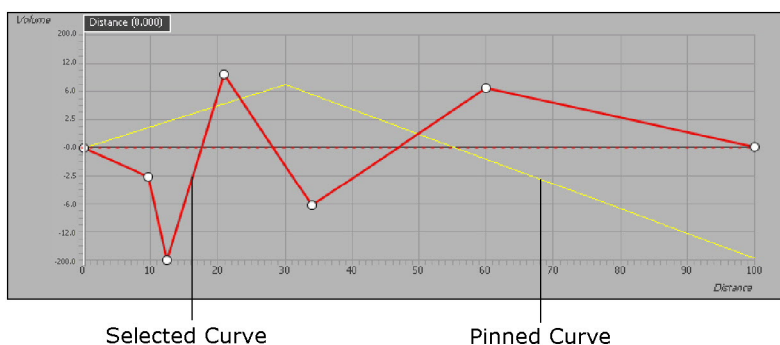
RTPC curves can be copied from one property to another or from one object to another. For more information on copying RTPC curves, refer to [Copying RTPC Curves](#).

Displaying Curves in the Graph View

In most of the graph views in Wwise, you can have several curves representing different paths or different relationships between game parameters and Wwise properties. When you select a curve from the curve list, it is displayed in the graph view. If the graph supports multiple curve display, such as the RTPC and Attenuation Editor graph views, you can Ctrl+click multiple curves to display all of them in the graph view simultaneously.

If you want a curve to remain displayed in the graph view without being selected, you can pin it to the graph view. When a curve is pinned, its outline remains in the graph view at all times. The curves control points, however, are not visible. A curve must be selected before its control points can be edited.

The following illustration shows the difference between selected and pinned curves in the RTPC graph view.



If several curves are displayed in the graph view, the units along the X and Y axes may disappear. This occurs when the game parameter or property units for the curves are different. For example, if you display a pitch and volume curve simultaneously, the units along the Y axis will disappear because one is in cents and the other is in decibels.



Note

If the scaling method of the curves displayed in the graph view is different, the units along the Y axis will not be displayed. For more information about the scaling method, refer to [Defining the Scaling Method of the Graph View](#).

To display curves in the graph view:

1. In graph view, select a curve from the curve list.

The curve is displayed in the graph view.

2. To display more curves, Ctrl+click the curves in the list.

The curves are displayed in the graph view. If the curves are measured using different units, the units along X and/or Y axes will disappear.

To pin a curve to the graph view:

1. In the curve list, click the pin icon for the curve that you want pinned to the graph view.

The pin icon turns blue and the curve is displayed in the graph view.

When you select another curve, the outline of the pinned curve will remain displayed in the graph view until you unpin it.

Related Topics

- [Specifying the Shape of the Curve Between Control Points](#)
- [Zooming and Panning the Graph View](#)
- [Defining the Scaling Method of the Graph View](#)

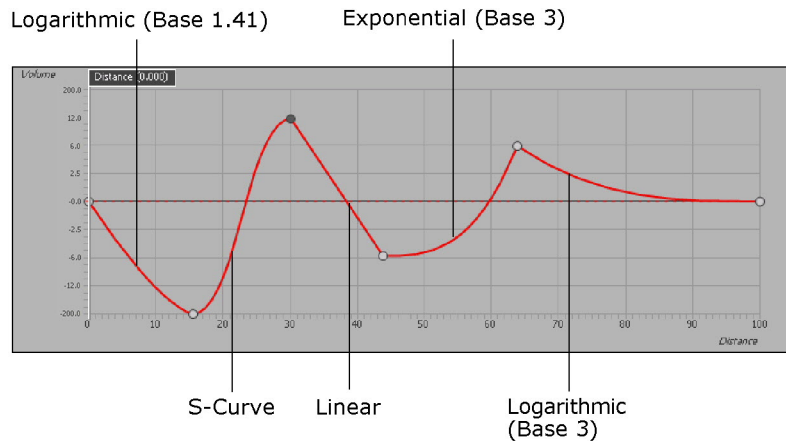
Specifying the Shape of the Curve Between Control Points

To give you greater control and flexibility over the curves in your project, Wwise allows you to define the shape of each curve segment. A curve segment is any part of the curve between two control points. You can choose from a variety of curve shapes, including linear, constant, logarithmic, exponential, and s-curve.



Note

While the curve tools provide flexibility, they require more processing and thus increase CPU usage.



To specify the shape of the curve between control points:

1. In the graph view, right-click a segment of the curve.

A shortcut menu is displayed

2. From the menu, select one of the following options:

Logarithmic (Base 3)

Sine (Constant Power Fade In) - this Sine curve shape only offers a constant power crossfade when used on the 'in' part of the curve.

Logarithmic (Base 1.41)

Inverted S-Curve

Linear

Constant

S-Curve

Exponential (Base 1.41)

Sine (Constant Power Fade Out) - this Sine curve shape only offers a constant power crossfade when used on the 'out' part of the curve.

Exponential (Base 3)

The selected curve shape is applied to the segment of the curve.

3. Continue to apply curve shapes to the other segments of the curve, as needed.

Related Topics

- [Displaying Curves in the Graph View](#)
- [Zooming and Panning the Graph View](#)
- [Defining the Scaling Method of the Graph View](#)
- [Adding Control Points](#)
- [Moving Control Points](#)
- [Deleting Control Points](#)

Chapter 43. Getting to Know the Timeline

Overview	860
Working with the Timeline for Positioning	863
Working with the Music Segment Editor Timeline	864

Overview

Several views in Wwise include a timeline so that you can use time-based information for different purposes in your project. The following views include a timeline:

- Music Segment Editor
- Position Editor
- Performance Monitor
- Game Sync Monitor

Although the timelines generally support many of the same functions, they do have their differences. The following table describes the unique purpose of each timeline.

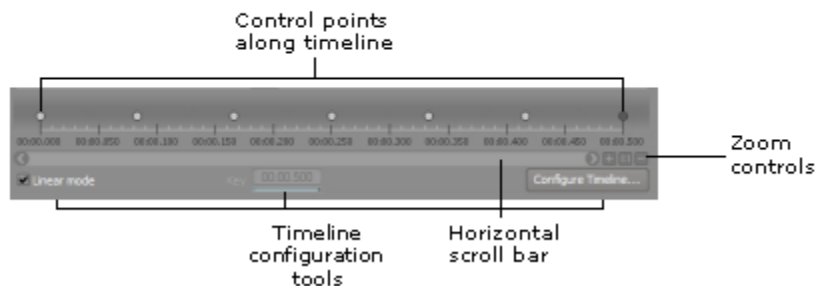
Use this timeline	To
Music Segment Editor	Arrange and synchronize interactive music components.
Position Editor	Define time periods for sound paths.
Performance Monitor	View time-based captured information from the Performance Monitor.
Game Sync Monitor	View time-based RTPC value changes for watched game objects.

When creating sound paths and monitoring performance, the timeline is used in tandem with a graph view. For more information on the graph view, refer to [Working with Control Points in the Graph View](#).

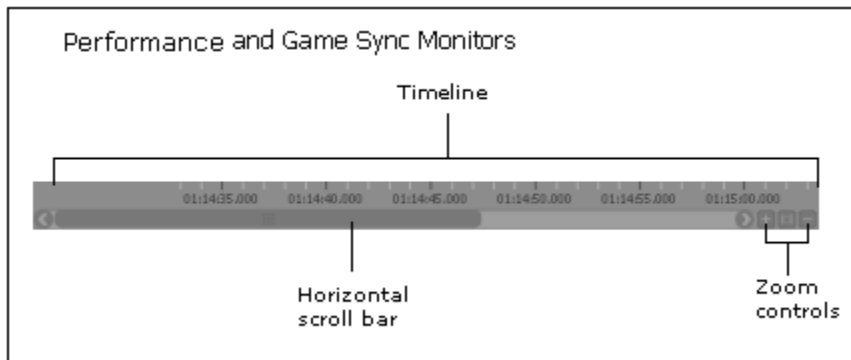
Describing the Elements of the Timeline

Because there are three distinct timelines in Wwise, each one will be discussed separately. In the Position Editor (3D User-defined), the control points added to the graph view are also plotted along a timeline. The timeline shows where the control points fall over time. By plotting the points in space and time, you can define the position of the sound as well as the time it takes for the sound to travel along the path. The following illustration shows the different elements in the Position Editor (3D User-defined) timeline.

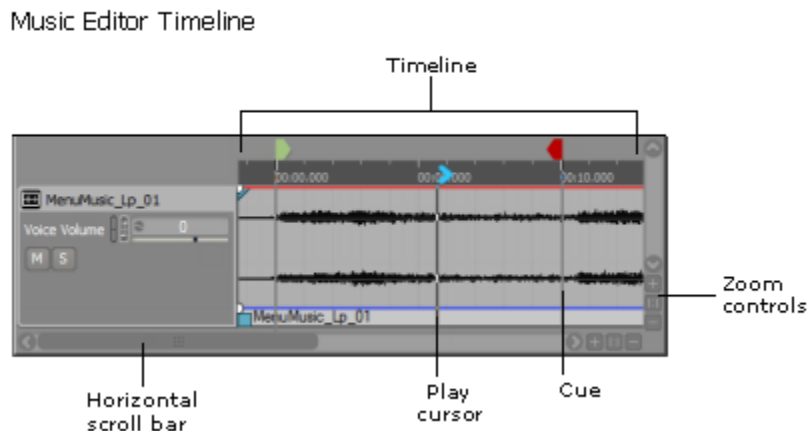
Position Editor Timeline



In the Performance Monitor, the timeline helps you locate each action and notification captured from the sound engine and logged in the Capture Log. The following illustration shows the different elements in the Performance Monitor timeline.



In the Music Segment Editor, music clips are displayed in tracks along a timeline. The timeline and the waveform itself help you locate the most appropriate sections of the music where transitions, state changes, and stingers could occur. Entry, exit, and any number of custom cues can be placed at these specific locations along the timeline.



Panning and Zooming the Timeline

You can use the pan and zoom tools to examine information more closely and to help you add control points or cues at very specific times. You can zoom in and out, and pan left and right. The timeline in Music Segment Editor contains these tools along both the X and Y axis.

All the pan and zoom controls have associated keyboard shortcuts.

To	Use this shortcut
Zoom in	Z+marquee selection

To	Use this shortcut
Zoom in – Centered at current mouse position	Ctrl+mouse wheel up
Zoom out – Centered at current mouse position	Ctrl+mouse wheel down
Reset zoom (when zoomed in)	Z+click
Pan view left (when zoomed in)	Shift +mouse wheel up
Pan view right (when zoomed in)	Shift + mouse wheel down
Pan view up (when zoomed in)	Ctrl+Shift + mouse wheel up.
Pan view down (when zoomed in)	Ctrl+Shift + mouse wheel down
Pan free hand	X+drag

For a complete list of shortcuts, refer to [Appendix B, *Shortcuts*](#).

To zoom in the timeline:

1. From the timeline toolbar, click the **Zoom In** icon.

The timeline zooms in towards the center of the timeline.

To zoom out the timeline:

1. From the timeline toolbar, click the **Zoom Out** icon.

The timeline zooms out from the center of the timeline.

To reset the pan and zoom in the timeline:

1. From the timeline toolbar, click the **Reset** icon.

The timeline is reset to show all control points.

To pan the timeline:

1. Zoom in the timeline.
2. Do one of the following:

To pan right, move the mouse wheel down.

To pan left, move the mouse wheel up.

To pan freehand, press x and drag the mouse in the timeline.

Related Topics

- [Configuring the Positioning Timeline](#)
- [Configuring the Music Segment Editor Timeline](#)
- [Extending and Shortening Music Tracks on the Timeline](#)

Working with the Timeline for Positioning

When creating spatial paths, you need to define the time it will take for the sound or motion object to travel along the path. The Position Editor timeline allows you to specify where each control point will fall over time. Like the graph view, you can zoom and pan in the timeline to help position control points more accurately. You can also configure the length of the timeline for each path you create.

- [Configuring the Positioning Timeline](#)

Configuring the Positioning Timeline

You can configure the properties and behaviors of the timeline. For example, you can specify the length of the timeline, or define the behavior of the control points on the timeline when new control points are added. The length of the timeline automatically defines the length of the selected path. You can configure the timeline so that each path that you've created is a different length.

When the timeline is in linear mode, you can only define the length of the timeline because the behaviors of the points on the timeline are pre-determined.

To configure the Timeline:

1. In the Position Editor (3D User-defined), click the **Configure Timeline** button.

The Timeline Configuration dialog box opens.

2. In the Length field, type the duration of the timeline in mm:ss.ms.



Note

The maximum length of the timeline is 59:59:999.

3. If you are changing the length of the timeline and it is in non-linear mode, select one of the following options:

Stretch proportionally to re-position existing control points to maintain their relative proportions to one another.

Preserve key values to maintain the positions of existing control points.



Note

If you shorten the length of the timeline using the Preserve key values option, some control points may be deleted. If points

are going to be deleted, a confirmation message is displayed prior to deletion.

4. In the Insert Key Every field, type the amount of time you want to add between the last existing control point and any new point that is inserted on the timeline.
5. Click OK to accept your changes.

The timeline is re-configured according to the new settings.

Related Topics

- [Panning and Zooming the Timeline](#)
- [Configuring the Music Segment Editor Timeline](#)
- [Extending and Shortening Music Tracks on the Timeline](#)

Working with the Music Segment Editor Timeline

The Music Segment Editor timeline is key to managing arrangements in your interactive music project. The timeline provides you with important feedback when you are working with segments, tracks, and clips.

When working with the Music Segment Editor timeline, you can do the following tasks:

- [Configuring the Music Segment Editor Timeline](#)
- [Extending and Shortening Music Tracks on the Timeline](#)

Configuring the Music Segment Editor Timeline

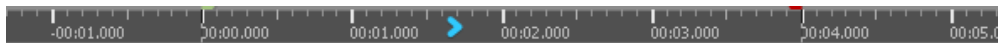
To optimally arrange your music, you can configure the timeline for the following:

- **Seconds** - Displays the timeline in seconds. As you zoom in to this view, the timeline displays the seconds in increased increments from tenths, hundredths, and thousandths of a second.
- **Bars and Beats** - Displays the timeline in bars and beats. Use this setting if you are working with musical material that must align with the musical meter. The increments on the timeline will be based on the Time Signature specified in the corresponding Property Editor.

To configure the timeline in the Music Segment Editor to Seconds:

1. Right-click on the timeline and select **Seconds**.

The timeline display is re-configured to seconds.



To configure the timeline in the Music Segment Editor to Bars and Beats:

1. Right-click on the timeline and select **Bars and Beats**.

The timeline display is re-configured to bars and beats.



Related Topics

- [Panning and Zooming the Timeline](#)
- [Configuring the Positioning Timeline](#)
- [Extending and Shortening Music Tracks on the Timeline](#)

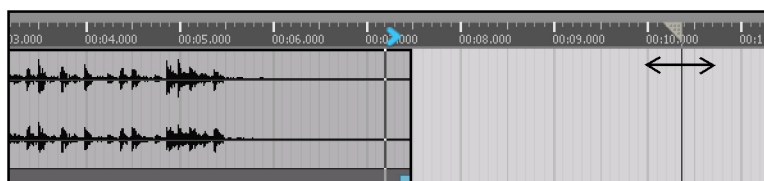
Extending and Shortening Music Tracks on the Timeline

When you are working with clips in the timeline you may want to adjust the length of the timeline.

To resize timeline in the Music Segment Editor:

1. 

Drag the End Cursor icon to the desired position along the timeline.



Drag end cursor icon to extend or shorten the segment tracks

Related Topics

- [Panning and Zooming the Timeline](#)
- [Configuring the Positioning Timeline](#)
- [Configuring the Music Segment Editor Timeline](#)

Chapter 44. Working with Searches, Queries, and References

Overview	867
Searching for Elements within Your Project	867
Finding the Project Elements that Reference a Particular Object	870
Working with Queries	872
Queries - Tips and Best Practices	880

Overview

A project in Wwise may contain thousands of sounds and hundreds of motion FX, containers, events, and other objects. As your project grows, it will become more and more important that you be able to find specific project elements quickly and easily. For this reason, Wwise provides you with three efficient and powerful ways to find elements within your project:

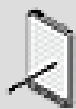
- **Searches** - A quick text-based search tool that displays matches as you type.
- **Queries** - A more powerful search tool that uses a series of conditions or criteria.
- **References** - A search tool that finds elements within your project that contain direct references to a particular object or element.

The Search tool is quick and easy to use. You simply have to type some text in the Search field and Wwise will look in the name and notes fields of every project element to find a match. All matches are automatically displayed in the Results list as you type.

The References tool is also very quick and easy to use. You simply have to right-click an object, select Find All References, and Wwise will create a list of objects and other project elements that contain direct references to the current object. The list of references will be displayed in the Reference view.

The Query tool, on the other hand, provides you with a more sophisticated and powerful search engine. Within a query, you can define a set of criteria, which allows you to search for very specific elements within your project. For example, you can create a query to find all streamed sound objects that use a prefetch length of 100 ms and then use the multi-edit command to change all of them to 150 ms. Queries do take a little more time to set up, but once created, they are very powerful and can be re-used at any point in your development cycle.

By using these two search tools, you will be able to find almost any element or series of elements within your project.



Note

If work units have been unloaded from your project, Wwise cannot search or query for elements within these work units.

Searching for Elements within Your Project

You can find any element in your project both quickly and easily using the Search tool. The Search tool is a simple text-based search. You simply have to

type some text in the Search field and Wwise will search through the entire project to find all elements with matching names or notes. If any matches are found, they are automatically displayed in the Results list.

The Search tool tries to create a match by looking at the start of each new word in the name or notes fields. It is important to understand that throughout Wwise, words are considered separate if they contain non alpha numeric characters, such as spaces, underscores, dashes, and so on, or a change in case. The Search tool takes this into consideration when looking for matches. The following table demonstrates a few different examples:

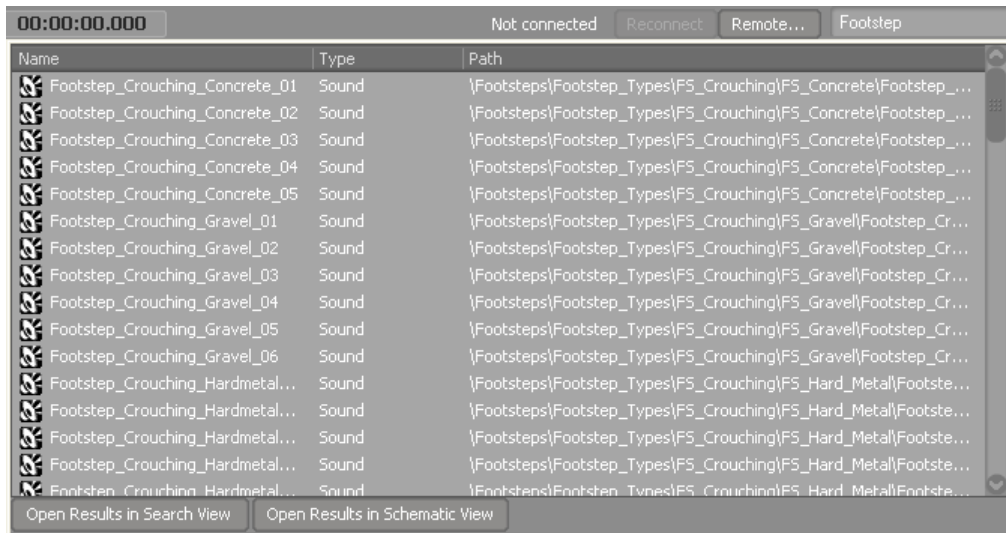
Search Field Text	Match	No Match
Big	Big_Car Big Car Bigcar VeryBigCar	Verybigcar
Small Car	Small Fast Car Small_Car Very small car	Smallcar Verysmallcar
mediumcar	mediumcar VeryMediumcar Very_Mediumcar	Medium Car

You can use the List View for performing searches, but you can also do a quick search using the Search field, which is located on the right side of the Wwise toolbar. From the list of results, you can load a project element into its corresponding editor. When using the List View, you can also right-click a selection of entries in the list to perform a variety of tasks, including multi-edit, convert, and delete.

To search for elements within your project:

1. In the Wwise toolbar, type the project element's name, notes, or a portion thereof, in the Search field.

As you type, a list of project elements with matching names/notes is displayed below the search field.



2. Do one of the following:

- Click **Open Results in List View** to display the complete list of results in the List View.
- Click **Open Results in Schematic View** to display the complete list of results in the Schematic view.
- Click the project element you were looking for to load it into its corresponding Editor.
- Use the arrow keys to scroll through the Results list and then press **Enter** to load the selected element into its corresponding Editor.



Note

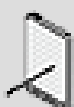
If the results are displayed in the List View, you can select an entry to automatically load it into the Transport Control for immediate playback. You can also right-click a selection of entries in the list to perform a variety of tasks, including multi-edit, convert, and delete.

Related Topics

- [Creating a Query](#)
- [Defining and Running a Query](#)
- [Deleting a Query](#)
- [Using Tables](#)

Finding the Project Elements that Reference a Particular Object

There may be cases during the development of your project where you want to find all the elements in your project that contain direct references to a particular object. For example, you may want to find which events reference a particular object or which SoundBanks reference a particular event. In Wwise, this is very easy using the Find all references command that is accessible from most of the right-click shortcut menus. All elements that reference a particular object are displayed in the Reference view, where you can easily open each project element, make changes, if necessary, and then move on.



Note

Only the project elements that contain a direct reference to a particular object or element, and not their child objects, will be displayed in the reference view.

The following table contains a list of all possible references for each element in your project.

Project Element	Possible References
Sound/Music Object	Audio Bus Motion Bus Effect ShareSet* Attenuation ShareSet* Game Parameter (via RTPC) State Group * - Custom instances of ShareSets are not included in the Reference view.
Switch Container	Switch Group State Group
Blend Container	Game Parameter (via RTPC on Blend Track)
Bus	Bus (via Ducking)
Event	Audio Object State Group State Switch Group Switch Trigger

Working with Searches, Queries, and References

Project Element	Possible References
Dialogue Event	Audio Objects State Group State
Music Object/Transition	Music Segment (via Transition Segment) Music Object (via Jump to playlist item) Music Object (via Source/Destination Transition Matrix)
Music Object/Stinger	Music Segment (via Stinger) Trigger
Music Switch Container	State Group Switch Group
SoundBank	Audio Object Music Object Work Unit Event
Soundcaster Session	Audio Objects Audio/Motion Bus Event
Switch Group	Game Parameter
State Group	States (via Transition Times)

In most cases, you will focus your search for references to one particular object, but you can also find all elements that reference a series of objects.



Note

The reference list does not get updated automatically, so if changes are made to the project, you will need to update the reference list manually by clicking the Refresh button.

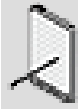
To find project elements that reference a particular object:

1. Do one of the following:

Right-click a project element or selection of project elements and select **Find all references** from the shortcut menu.

Open the Reference view by pressing Shift+F3, and then drag one object or a selection of objects to the References to: field.

Open the Reference view by pressing Shift+F3, click the Browse button beside the References to: field, select the project element, and then click OK.



Note

If you press Shift+F3 while an object or element is selected in the Project Explorer, Property Editor, or the Reference View, Wwise automatically searches for objects that contain references to the selected object.

A list of project elements that contain a reference to the selected object(s) is displayed in the Reference view.

Related Topics

- [Searching for Elements within Your Project](#)
- [Working with Queries](#)

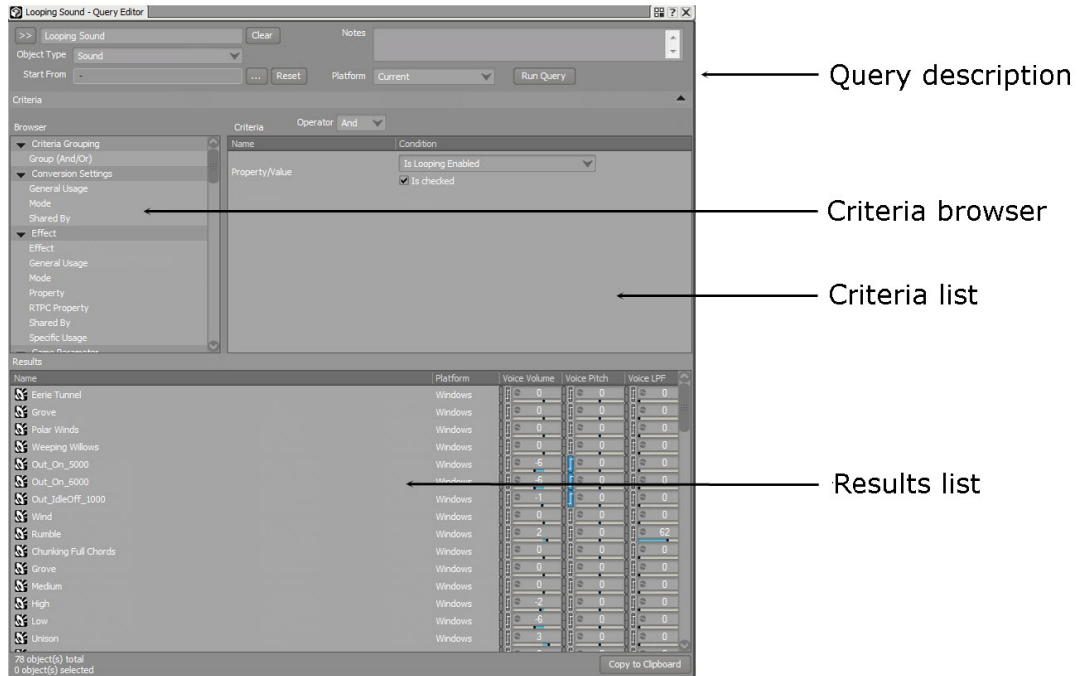
Working with Queries

When you need to find a particular object in your project and the Search tool does not meet your needs, you can create a query in the Query Editor. The Query Editor is a sophisticated and powerful tool that can help you search through a project and find any object you're looking for. Your queries can be as broad or as specific as required. You can also save and reload queries to make your workflow more efficient. After running a query, you can edit all objects within the Results pane at once. For example, you can use the Multi-Editor to turn streaming on or off, or change the conversion settings for the entire group of objects in the Results pane.

For example, you could create and run queries to do the following:

- To locate all sounds in your project that begin with the word sword, create a query for sounds with sound type SFX and the name sword*.
- To locate all music segments routed through a given bus, create a query for music segments with a selected output bus.
- To locate all voices with a low pass filter property higher than 10, create a query for sounds with sound type Voice and the property/value low pass filter value greater than 10.

The Query Editor is composed of the following sections:



Creating a Query

When you create a query in Wwise, you set up a unique search that will be used to find objects in a project. After you name your query, it is saved for future use.

You can create queries in two places in Wwise:

- Queries tab of the Project Explorer
- Within the Query editor

To create a new query:

1. In the Project Explorer, switch to the **Queries** tab.
2. Do one of the following:

Select a work unit of virtual folder and click the **Query** icon in the Project Explorer toolbar.

Right-click a work unit or virtual folder and select **New Child > Query** from the shortcut menu.

A new query is added to the selected work unit or virtual folder.

3. Replace the default name with one that best represents the query, and then press **Enter**.



Note

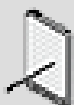
Each query name must be unique, and may not include the following characters: ‘:<>*?”\|.’ You can rename a query at any time by right-clicking it, selecting *Rename*, and typing a new name.

To create a new query from within the Query Editor:

1. In the Query Editor, click *New...*

The New Query dialog opens.

2. Select the work unit in which you want to create the query.
3. Type a name for the query.



Note

Each query name must be unique, and may not include the following characters: ‘:<>*?”\|.’ You can rename a query at any time by right-clicking it, selecting *Rename*, and typing a new name.

4. Click *OK*.

A new query is created.

Related Topics

- [Defining and Running a Query](#)
- [Creating Advanced Queries Using Criteria Groups](#)
- [Using Factory-Defined Queries](#)
- [Deleting a Query](#)
- [Searching for Elements within Your Project](#)

Defining and Running a Query

When you run a query for the first time, you decide which criteria will be used for searching through the project. These criteria are saved automatically so you can save time when you run the same query again in the future.



Note

Query results take into account object-level properties that are not currently in use because they are being overridden by a parent. If you encounter unexpected results in the Query Editor, you can look for hidden object-level properties by temporarily enabling the Override parent option in the Property Editor.

To define the contents of a query:

1. Load a query into the Query Editor.
2. Type any additional information about the query in the Notes field.
3. From the Object Type list, select the type of object or other project element you want to search for. If you do not want to search for a specific type of object, select **All Objects**.
4. To specify a location in the hierarchy from which your query will begin, do the following:

Next to the Start From Here box, click the **Browse** button (...).

The Project Explorer - Browser dialog box opens.

Navigate to the folder where you want Wwise to begin searching.

Click **OK**.



Note

To clear the starting location you have selected, click **Reset**.

5. From the Platform list, select one of the following options:

All to search for objects that are used on any platform included in the project.

Current to search for objects that are used on the currently selected platform.

6. In the browser list, select each criterion on which you want to base your query. Do the following:

Double-click a criterion to load it into the Criteria list.

Type or select a condition for the criterion. You can use the wildcard operator (*) to replace part of a word when you are typing a criterion.



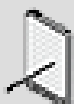
Note

To delete a criterion, right-click it and select **Remove**.

7. From the Operator list, select one of the following:

And to return results for which all your criteria are true.

Or to return results for which at least one of your criteria are true.



Note

If you need to create a more sophisticated query that contains both operators, you can create a criteria group. For more information on criteria groups, refer to [Creating Advanced Queries Using Criteria Groups](#).

8. Click **Run Query**.

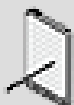
The query results are displayed. You can sort these results by clicking each column heading.



Tip

When you select an entry in the Results list, it automatically gets loaded into the Transport Control and is ready for playback. You can right-click a selection of entries in the Results list to access a series of commands, such as Edit, Multi-edit, Show in Schematic View, and Convert. You can double-click any entry in the Results list to display it in the Property Editor. You can also drag project elements from the Results list to the Project Explorer. For example, you can drag one or more sound objects from the Results list into an Actor-Mixer or container within the Project Explorer.

9. To keep the results of your query for use in another application, click **Copy to Clipboard**.
10. To clear your criteria and search results, click **Clear**.



Note

In Wwise, objects have inherent properties that persist even if these are replaced by those of parent objects. If you are finding

false positives in your results, that is, results that do not contain the criteria for which you were searching, these objects might have inherent properties that are being overridden with the Override parent option.

Related Topics

- [Creating a Query](#)
- [Creating Advanced Queries Using Criteria Groups](#)
- [Using Factory-Defined Queries](#)
- [Deleting a Query](#)
- [Searching for Elements within Your Project](#)

Creating Advanced Queries Using Criteria Groups

There may be situations where a basic query that uses one operator is not specific enough to find exactly what you are looking for. In these cases, you can create more advanced searches using criteria groups. Criteria groups are a subset of criteria with their own separate operator control. This allows you to create a query with different types of operators. For example, you can create a query that searches for all objects with names that start with the letter P AND that also contain either the number 1 OR 0.

The screenshot shows the Query Editor interface. At the top, there is a 'Criteria' section with an 'Operator' dropdown set to 'And'. Below this is a table with two columns: 'Name' and 'Condition'. The first row has 'General/Name' in the 'Name' column and 'P*' in the 'Condition' column. Below this is a 'Criteria Grouping/Group (And/Or)' section with a dropdown set to 'Or'. To the right of this dropdown is a note: 'Note: This operator applies to sub-criteria only. To add a sub-criterion, select this row before double-clicking on the desired criterion type in the list on the left.' Below the grouping section are two rows, each with 'General/Name' in the 'Name' column and '*0*' and '*1*' in the 'Condition' column respectively. A bracket on the left side of these two rows indicates they are grouped together under the 'Or' operator.

You can use one or more criteria groups within the same query to find specific objects or other elements within your project. Criteria groups can even be nested within other criteria groups to create very sophisticated queries.

To create a criteria group within a query:

1. Load a query into the Query Editor.

2. In the Browser list, double-click the **Group (And/Or)** option under the Criteria Grouping section.

A new criteria group is added to the Criteria list.

3. In the browser list, add one or more criteria to the criteria group by doing the following:

Select the criteria group in the Criteria list.

Double-click a criterion to add it into the Criteria group.

Type or select a condition for the criterion. You can use the wildcard operator (*) to replace part of a word when you are typing a criterion.



Note

To re-order the criteria within a group, simply drag the criterion to the new location. Once a criterion has been added to a group, it can't be dragged outside the group.

4. From the Criteria Group Operator list, select one of the following:

And to return results for which all your criteria within the group are true.

Or to return results for which at least one of your criteria within the group are true.



Note

To add another criterion outside the group, click outside the group to deselect it, and then double-click the criterion in the Browser list.

5. Click **Run Query**.

The query results are displayed. You can sort these results by clicking each column heading.



Tip

When you select an entry in the Results list, it automatically gets loaded into the Transport Control and is ready for playback. You can right-click a selection of entries in the Results list to access a series of commands, such as Edit, Multi-edit, Show in Schematic View, and Convert. You can double-

click any entry in the Results list to display it in the Property Editor. You can also drag project elements from the Results list to the Project Explorer. For example, you can drag one or more sound objects from the Results list into an Actor-Mixer or container within the Project Explorer.

6. To keep the results of your query for use in another application, click **Copy to Clipboard**.
7. To clear your criteria and search results, click **Clear**.

Related Topics

- [Defining and Running a Query](#)
- [Using Factory-Defined Queries](#)
- [Deleting a Query](#)

Using Factory-Defined Queries

Wwise comes with pre-defined factory presets for many common queries. You can save time by using these instead of defining queries from scratch.

To run a factory-defined query:

1. In the Project Explorer, switch to the Queries tab.
2. From the Factory Queries list, double-click a query.

The query is loaded into the Query Editor.

3. Type or select a condition for the criteria as required. You can use the wildcard operator (*) to replace part of a word when you are typing a criterion.
4. Click **Run Query**.

The query results are displayed. You can sort these results by clicking each column heading.



Tip

When you select an entry in the Results list, it automatically gets loaded into the Transport Control and is ready for playback. You can right-click a selection of entries in the Results list to access a series of commands, such as Edit, Multi-edit, Show in Schematic View, and Convert. You can double-click any entry in the Results list to display it in the Property Editor. You can also drag project elements from the Results list to the Project Explorer. For example, you can drag one or

more sound objects from the Results list into an Actor-Mixer or container within the Project Explorer.

5. To keep the results of your query for use in another application, click **Copy to Clipboard**.
6. To clear your criteria and search results, click **Clear**.

Related Topics

- [Creating a Query](#)
- [Creating Advanced Queries Using Criteria Groups](#)
- [Deleting a Query](#)
- [Searching for Elements within Your Project](#)

Deleting a Query

You may want to delete a query that you no longer need.

To delete a query:

1. In the Project Explorer, switch to the Queries tab.
2. Right-click the query that you want to delete, and select **Delete Selection**.

The selected query is deleted.



Note

If you delete a query by mistake, you can undo the delete by pressing **Ctrl+Z**, or by clicking **Edit > Undo**.

Related Topics

- [Creating a Query](#)
- [Creating Advanced Queries Using Criteria Groups](#)
- [Using Factory-Defined Queries](#)
- [Defining and Running a Query](#)
- [Searching for Elements within Your Project](#)

Queries - Tips and Best Practices

Many options are available to you when you create queries in Wwise. By using certain strategies when designing queries, you can get the precise results you need, quickly and consistently. The following are some strategies you might consider when building queries for use in your projects.

Object Type Details

Many criteria in the browser list have conditions that allow you to further narrow down a query. However, these conditions will vary depending on the object type you select. For example, consider the following three cases for the Property Value criteria:

- If you select “Audio Source” as your object type, a list of conditions including Bit Depth and Sample Rate is available for you to choose from.
- If you select “Event” as your object type, no conditions are available as this is not a valid criteria for this object type (Events in Wwise don't have properties).

Therefore, your choice of object type is very important as it determines the conditions you can use for your queries.

Speeding Up Queries

A few simple choices can make your queries run much faster.

- Specify an object type.
- Specify a platform, if you are looking for something platform-specific.
- Use **Start From Here** to choose a location as far down in the hierarchy as possible.

Using Wildcards in Queries

Wildcards are symbols that add flexibility to a keyword search by extending the parameters of a search word. You can use the wildcard operator, which is the asterisk (*) in Wwise, to replace part of a word when you are trying to find different project elements that contain, start with, or end with certain letters or numbers. For example, if you want to find all objects with LOOP in the name, you should type *LOOP* within the condition text box. If, on the other hand, you want to find all objects that start with the letters LOOP, you should type LOOP* within the condition text box. Finally, if you want to find all objects that end with the letters LOOP, then you should type *LOOP within the condition text box.

Chapter 45. Using Presets

Overview	883
Using Presets	883

Overview

Presets are a specific set of property values related to an object or effect that are saved into a special file so that they can be re-used at a later time within the same project. By using presets, you don't have to recreate particular property setups that you want to re-use for other objects in your project. All you have to do is set up the property values once, save the preset, and then apply it to the other objects in your project. You will save time and effort by working more efficiently.



Note



Unlike ShareSets, the property values in a preset cannot be shared across objects.

Using Presets

The preset icons appear in the title bar of each view where presets can be saved and loaded.



The Load and Save Preset icons are shown in the following table:

Icon	Name
	Load Preset
	Save Preset

Saving Presets

You can save presets for any of the following elements in Wwise:

- Property values and settings for objects and sources within the Actor-Mixer and Interactive Music hierarchies
- Positioning values
- Attenuation settings
- Effect values

When you save presets, Wwise saves every value on every tab within the view. When a preset is saved, it is grouped according to one of the following categories:

- Actor-Mixer
- Random or Sequence Container
- Switch Container
- Blend Container
- Sound SFX/Voice
- Motion FX
- Audio or Motion Source Plug-in
- Music Switch Container
- Music Playlist Container
- Music Segment
- Music Track
- Positioning (3D User-defined)
- Positioning (2D)
- Attenuation
- Effects

When you open the Save Preset dialog box, the presets will be filtered to show only those presets that are in the same category.

To save a preset:

1. From the view title bar, click the **Save Preset** icon.

The Save Preset dialog box opens.

2. In the Name field, type the name of the preset.
3. In the Notes field, type any information that further describes the preset.
4. Click **Save** or press **Enter**.

The preset is saved and can be re-used at any time within the same project.

Related Topics

- [Loading Presets](#)
- [Deleting Presets](#)

Contents of a Preset

Since you can save presets for object properties at different levels within the project hierarchy, it is important to know what elements are saved with each preset. The following table describes all the presets along with the information that is saved.

Preset	Location in Hierarchy	Contents
Object/Source Properties	Top-level object	All property values on every tab within the Property Editor.

Preset	Location in Hierarchy	Contents
	Child object	All property values on every tab within the Property Editor. If the Override parent option is selected, the preset will contain all property values on every tab within the Property Editor including its own output, effect, positioning, playback limit and priority, and volume threshold settings.
Effect	-	All property values of the effect.
Positioning (2D)	-	The location of the point source.
Positioning (3D User-defined)	-	All positioning paths, settings, and values in the Position Editor (3D User-defined).
Attenuation	-	All curves and all attenuation settings.

Loading Presets

After you have saved your presets, you can apply them to other objects or effects within your project. When a preset is saved, it is grouped according to one of the following categories:

- Actor-Mixer
- Random or Sequence Container
- Switch Container
- Blend Container
- Sound SFX/Voice
- Motion FX
- Audio or Motion Source Plug-in
- Music Switch Container
- Music Playlist Container
- Music Segment
- Music Track
- Positioning (3D User-defined)
- Positioning (2D)
- Attenuation
- Effect

When you open the Load Preset dialog box, the presets will be filtered to show only those presets that are in the same category.

To load a preset:

1. From the view title bar, click the **Load Preset** icon.

The Load Preset dialog box opens.

2. Select a preset from the Preset list.
3. Click **Load** or press **Enter**.

The preset is applied to the object or effect.

Related Topics

- [Saving Presets](#)
- [Deleting Presets](#)

Deleting Presets

If you no longer need a preset, you can delete it.

To delete a preset:

1. From the view title bar, click the **Load Preset** icon.

The Load Preset dialog box opens.



Tip

You can also delete presets from the Save Preset dialog box.

2. In the Preset list, click the preset that you want to delete.

The Delete button becomes enabled.

3. Click **Delete**.

The preset is removed from the Preset list and deleted from your project.

Related Topics

- [Saving Presets](#)
- [Loading Presets](#)

Chapter 46. Using a Control Surface

Overview	888
Connecting a Control Surface Device to Wwise	888
Creating a Control Surface Session	889
Understanding Control Surface Bindings	889
Creating Control Surface Bindings	890
Understanding Control Surface View Groups	893
Handling Conflicts in Control Surface Sessions	895
Using the Control Surface Toolbar	895

Overview

Control Surface devices can be used to control Wwise functions or project properties. Wwise supports the MIDI protocol and the Mackie protocol. It is also possible to use Wwise with iOS and Android based devices with the TouchOSC application in pair with the TouchOSC Bridge, which does support MIDI.

Connecting a Control Surface Device to Wwise

Wwise supports two types of protocols for control surfaces:

- MIDI Protocol
- Mackie HUI MIDI Mapping Protocol (MCU Pro)

Before you begin:

- Ensure your device is physically connected to your computer
- Ensure your device is turned ON
- Ensure you have the proper drivers installed for your device



Note

Wwise does not support the Open Source Control (OSC) protocol natively. But applications such as TouchOSC have MIDI support and can be used with TouchOSC Bridge.

To connect your device to Wwise:

1. From the **Project** menu, select **Control Surface Devices**.
2. Click the **Add** button.
3. Give a name to your device.
4. Click **OK**.

The device is added to the list.

5. In the **Receive From** column, select the MIDI IN device.

The **Connected** message appears.

6. In the **Send To** column, select the MIDI OUT device.

The **Connected** message appears.

7. Click **Close**.

The device is now ready to use.



Note

The Control Surface devices settings are stored locally on the computer.

Creating a Control Surface Session

Control Surface Sessions define how the hardware controls are attached to Wwise functions or project properties. A Control Surface Session defines a list of bindings. Each binding attach a hardware control (button, slider, knob, key, etc) to a Wwise element (property or command).



Note

The Control Surface Sessions are stored within the project, and can be used on any computer where the project is being used.

To create a Control Surface Session:

1. From the **View** menu, open the **Control Surface Bindings** view (Ctrl+Shift+Q).
2. Click the [>>] button to open the selector menu.
3. Select **New...**
4. Enter a name for the session.
5. Press **OK**.

The session is created and loaded.

Multiple Control Surface Sessions can be created in a single Wwise Project for different reasons:

- Different users of the Wwise projects using different hardware devices.
- Different usage scenarios.
- Different users with different needs or preferences.



Note

Although you can have multiple sessions in the project, only one session can be active at a time.

Understanding Control Surface Bindings

A Control Surface binding attaches a hardware control (button, slider, knob, key, etc) to a Wwise element (property or command).

Every binding has three elements:

- **Property/Command:**
 - Object Property: A property name to modify, on the targeted object.
 - Object Command: A command or action to launch on the targeted object.
 - Global Command: A command or action to launch, globally (not targeting an object).
- **Object/Index:** Specify the object to target.
- **Controller Assignment:** Identify the hardware control element with a MIDI message ID.

Bindings are stored inside three different kinds of groups, that each have a different mechanisms to define the targeted object:

- **Global:** The targeted object is specified directly in the binding.
- **Current Selection:** The targeted object is the latest selected object in Wwise.
- **View Groups:** The target object is defined by the view the binding group is being loaded. An index is specified for each object loaded in the view.

Related Topics

- [Creating a Binding for a Keyboard Shortcut](#)
- [Creating a Binding to Modify a Specific Object Property Value](#)
- [Creating a Binding to Modify the Current Selection](#)

Creating Control Surface Bindings

Bindings can be created for multiple scenarios:

- [Creating a Binding for a Keyboard Shortcut](#)
- [Creating a Binding to Modify the Current Selection](#)
- [Creating a Binding to Modify a Specific Object Property Value](#)

Creating a Binding for a Keyboard Shortcut

Global Bindings can be used to trigger the same global commands that are also accessible in the Keyboard Shortcut manager. You can have a global command that is both triggered by the Keyboard Shortcut Manager and by a Control Surface Session's Binding.

To create a global binding in the Control Surface Bindings view:

1. Select the Global group.
2. Click the **Add & Learn Binding** button.

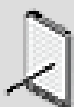
The binding entry is added and the **Learn** button is activated.

Properties/Commands that can be selected via the UI become green.

3. Click the [>>] button to select the Command.
4. Select **Global commands...** from the menu
5. Browse for the global command to launch.
6. Click OK.
7. Use the hardware control desired for the binding.

The **Learn** button is deactivated.

The binding is created and ready.



Note

Bindings for **Global Commands** can be created in other groups, such as the **Current Selection** group or the **View** groups

Creating a Binding to Modify a Specific Object Property Value

A Control Surface binding can be created to target a specific object property value in the Project. This can be useful to attach a controller assignment to objects such as Game Parameters.

To create a binding that controls a Game Parameter simulation value:

1. Select the **Global** group.
2. Click the **Add & Learn Binding** button.

The binding entry is added and the **Learn** button is activated.

Properties/Commands that can be selected via the UI become green.

3. Select the Property/Command by either:
 - Clicking on the game parameter's simulation value in the Transport Control.
 - Clicking the [>>] button and browse for the Property:
 - Select **Object Properties...** from the menu
 - Browse for **Games Syncs > Game Parameter > Simulation Value.**
 - Click OK.
4. Use the hardware control desired for the binding.

The **Learn** button is deactivated.

The binding is created and ready.

Creating a Binding to Modify the Current Selection

The Control Surface Session can define bindings that attach the property values of the current selection to hardware controller controls (sliders, knobs, buttons).

A typical setup would be to assign four sliders of a control surface to:

- Voice Volume.
- Voice Pitch.
- Voice Low-pass Filter.
- Voice High-pass Filter.

After the bindings are created, the four sliders will automatically be attached to these properties for the selected objects. If you have motorized faders (ex: The Mackie Control Universal Pro), the faders will automatically move to the current value as the current selection changes.

To create a binding that modifies the Voice Volume of the current selection:

1. Select the **Current Selection** group.
2. Click the **Add & Learn Binding** button.

The binding entry is added and the **Learn** button is activated.

Properties/Commands that can be selected via the UI become green.

3. Select the Property/Command by either:
 - Clicking on the Voice Volume in the Property Editor.
 - Clicking the [**>>**] button and browse for the Property:
 - Select **Object Properties...** from the menu
 - Browse for **Audio > General Settings > Voice Volume**.
 - Click OK.
4. Use the hardware control desired for the binding.

The **Learn** button is deactivated.

The binding is created and ready.

The current selection binding can also be used to trigger commands on the current selection. For example, you could map an array of buttons to:

- Mute

- Solo
- Play
- Stop

To create a binding that mutes the current selection:

1. Select the **Current Selection** group.
2. Click the **Add & Learn Binding** button.

The binding entry is added and the **Learn** button is activated.

Properties/Commands that can be selected via the UI become green.

3. Select the Property:
 - Click the [**>>**] button.
 - Select **Object commands > Mute** from the menu.
4. Use the hardware control desired for the binding.

The **Learn** button is deactivated.

The binding is created and ready.

If you have controller buttons that light-up, the light will turn on when muted and turn off when not muted.

Understanding Control Surface View Groups

View Groups can be used to bind the content of a specific view (objects) to the Binding Group content. You could, for example, map the objects of the Mixing Desk to a Control Surface.

A View Group contains bindings that are associated with objects of a view, identified by their index in the view.

Here are the views supporting View Groups and how they assign the indexes:

- **Mixing Desk:** one index per vertical strip.
- **List View, Query Editor, Reference View, Master-Mixer Console:** one index per row.
- **Property Editor, Effect Editor, Source Editor, Modulator Editor:** index 1 only, for the current object.
- **Soundcaster:** one index per module.

To create a View Group:

1. In **Control Surface Bindings View**, click the **View Groups** folder

2. Click the **Add Group** button.
3. Name your group.
4. Press OK.
5. Create bindings in the group using the **Add & Learn Binding** button.

For more information, refer to [Creating Control Surface Bindings](#).

Here is an example of a Control Surface View Group that can be used inside the Mixing Desk, List View or Soundcaster:

- **Mixing View Group:**
 - Binding: Property:Voice Volume - Index:1 - Key:MyController Ch.1 CC 0
 - Binding: Property:Voice Volume - Index:2 - Key:MyController Ch.1 CC 1
 - Binding: Property:Voice Volume - Index:3 - Key:MyController Ch.1 CC 2
 - Binding: Property:Voice Volume - Index:4 - Key:MyController Ch.1 CC 3
 - Binding: Command:Solo - Index:1 - Key:MyController Ch.1 CC 32
 - Binding: Command:Solo - Index:2 - Key:MyController Ch.1 CC 33
 - Binding: Command:Solo - Index:3 - Key:MyController Ch.1 CC 34
 - Binding: Command:Solo - Index:4 - Key:MyController Ch.1 CC 35
 - Binding: Command:Mute - Index:1 - Key:MyController Ch.1 CC 64
 - Binding: Command:Mute - Index:2 - Key:MyController Ch.1 CC 65
 - Binding: Command:Mute - Index:3 - Key:MyController Ch.1 CC 66
 - Binding: Command:Mute - Index:4 - Key:MyController Ch.1 CC 67

This View Group will map:

- 4 hardware sliders to Voice Volume of 4 objects.
- 4 hardware buttons to Solo of 4 objects.
- 4 hardware buttons to Mute of 4 objects.

To associate a View Group with the Mixing Desk:



Note

You must have an active Control Surface Session, and at least one view group created for these steps.

1. From the **View** menu, open the **Mixing Desk** view (Ctrl+Shift+M).
2. Click the [**>>**] button at the upper right corner of the Mixing Desk, on the title-bar.
3. From the selector menu, select the View Group to use.

The View Group is loaded.

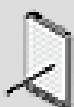


Note

View groups loaded in views are also shown in the Wwise toolbar. They can be activated and deactivated from the toolbar.

Handling Conflicts in Control Surface Sessions

It is possible that multiple active bindings enters in conflicts with their Controller Assignments. The Control Surface system does not allow to have multiple bindings with the same key that are loaded simultaneously.



Note

Bindings from a Control Surface Session are loaded from top to bottom. When a conflict arise, only the first conflicting binding will be loaded. Priority is given to the first binding found.

When a conflict is found:

- A yellow triangle is shown beside the group name in the Wwise toolbar.
- A yellow message is shown beside the binding in the **Control Surface Session** view.

There are multiple ways to handle conflicts:

- Re-order the bindings in the **Control Surface Session** view, to change priority.
- Use the shortcut menu on the binding entries to resolve the conflict.
- Assign a new controller assignment for one of the conflicting bindings.



Tip

In some scenarios, it might be useful to assign the same controller assignment to different bindings on the same group. For example, you could create an Effect View Group that binds properties of different effects. Because the Effect Editor can only load one effect at the time, the bindings will never conflicts.

Using the Control Surface Toolbar

The Control Surface toolbar shows the following elements:

Interface Element	Description
Back Navigation button	Navigates to the previous selected object (Backspace).

Interface Element	Description
Next Navigation button	Navigates forward in navigation list (Shift +Backspace)
Current Selection Object	Shows the current selected object.
Last Property Changed	Shows the last object, property and value being modified with a Control Surface's binding.
Loaded Binding Groups	Shows which groups are active and inactive. Clicking the group names can change the group activity. <ul style="list-style-type: none">• Active:blue• Inactive:gray



Note

The Control Surface toolbar is only visible when there is a **Control Surface Session** loaded in the **Control Surface Bindings** view.



A. Regular Expression Quick Reference Guide	899
B. Shortcuts	901
Zoomable Editor	902
Attenuation Editor	902
Audio File Management	902
Contents Editor	903
Game Object 3D Viewer	903
Game Profiler	904
Global - Contextual (Active view and selected object)	904
Music Segment Editor	904
Position Editor (User-defined)	905
Project Explorer	905
RTPC Graph View	905
Schematic View	905
Soundcaster	906
Transport Control	906

Appendix A. Regular Expression Quick Reference Guide

Anchors	
^	Start of string, or start of line in multi-line pattern
\$	End of string, or end of line in multi-line pattern
\b	Word boundary
\B	Not word boundary

Character Classes	
\s	White space
\S	Not white space
\d	Digit
\D	Not digit
\w	Word
\W	No Word
\x	Hexadecimal digit (Used for matching hex code characters, such as "\xA9", as in "0xA9", for the copyright symbol; or "\x5B", as in "0x5B", for the open square bracket.)

Quantifiers	
*	White space
+	Not white space
?	Digit
{3}	Exactly 3
{3,}	3 or more
{3,5}	3, 4, or 5

Escape Sequences	
\	Escape following character $\wedge [\. \$ \{ * (\ +)] ? < >$

Special Characters	
\n	New line
\r	Carriage return
\t	Tab

Groups and Ranges	
.	Any character except new line (\n)
\r	Carriage return

Regular Expression Quick Reference Guide

Groups and Ranges	
(a b)	a or b
(...)	Group
(?:...)	Passive (non-capturing) group
[abc]	Range (a or b or c)
[^abc]	Not (a or b or c)
[a-q]	Lower case letter from a to q
[A-Q]	Upper case letter from A to Q
[0-7]	Digit from 0 to 7

String Replacement	
\$n	nth non-passive group
\$2	"xyz" in /^(abc(xyz))\$/
\$1	"xyz" in /^(?:abc)(xyz)\$/
\$`	Before matched string
\$'	After matched string
\$+	Last matched string
\$&	Entire matched string

Appendix B. Shortcuts

Table of Contents

Zoomable Editor	902
Attenuation Editor	902
Audio File Management	902
Contents Editor	903
Game Object 3D Viewer	903
Game Profiler	904
Global - Contextual (Active view and selected object)	904
Music Segment Editor	904
Position Editor (User-defined)	905
Project Explorer	905
RTPC Graph View	905
Schematic View	905
Soundcaster	906
Transport Control	906

Many of the commands or operations within Wwise have been mapped to a key or key combination on your keyboard. You can use these keyboard shortcuts instead of the mouse to perform any of these actions or commands.

Most of the commands (found in the following pages) can be remapped by using the Keyboard Shortcut Manager.

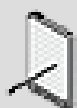
Zoomable Editor

To	Use this shortcut
Zoom in	Z+marquee selection
Zoom in horizontally – centered at current mouse position	Ctrl+mouse wheel up
Zoom out horizontally – centered at current mouse position	Ctrl+mouse wheel down
Zoom in vertically – centered at current mouse position	Ctrl+Shift+mouse wheel up
Zoom out vertically – centered at current mouse position	Ctrl+Shift+mouse wheel down
Reset pan and zoom (when zoomed in)	Z+click
Pan view up (when zoomed in)	Mouse wheel up
Pan view down (when zoomed in)	Mouse wheel down
Pan view left (when zoomed in)	Shift+mouse wheel up
Pan view right (when zoomed in)	Shift+mouse wheel down
Pan freehand (when zoomed in)	X+drag
Reset pan (when zoomed in)	X+click
Insert a point	Double-click
Move selected point	Arrow keys
Move point more accurately	Shift+drag
Select non-contiguous points	Ctrl+click
Select all points	Ctrl+A
Move selection to next point	Tab
Move selection to previous point	Shift+Tab
Lock selection to X or Y axis	Alt+Drag

Attenuation Editor

Refer to [Zoomable Editor](#).

Audio File Management



Note

These shortcuts work from Windows® folders.

To	Use this shortcut
Create a new Sound Voice	Shift+drag
Import SFX or Music audio files without opening the Audio File Importer.	Ctrl+drag
Import Sound Voice audio files without opening the Audio File Importer	Ctrl+Shift+drag
Force replace a Sound SFX or Music source	Alt+drag
Force replace a Sound Voice	Shift+Alt+drag

Contents Editor

To	Use this shortcut
Move selection up/down	Up/Down arrow keys
Add objects to selection (up/down)	Shift+Up/Down arrow keys
Move selection to next control on the right	Tab
Move selection to next control on the left	Shift+Tab
Move selection to next control below	Ctrl+Tab
Move selection to next control above	Ctrl+Shift+Tab

Game Object 3D Viewer

The following shortcuts are only available when the Game Object 3D Viewer is active.

To	Use this shortcut
Open the Game Object 3D Viewer Settings dialog box	V
Reset the camera to its default position	Ctrl+R
Set the camera to follow the selected game object during capture.	Ctrl+F
Frame All game objects and listeners in the view.	Ctrl+A
Display the Game Object 3D Viewer Filter dialog box	Alt+F
Navigate forward (in First Person or Listener camera)	W
Navigate backward (in First Person or Listener camera)	S
Navigate Right (in First Person or Listener camera)	A
Navigate left (in First Person or Listener camera)	D
Accelerate movement within the viewer	Shift
Switch to Camera User 1	Ctrl+1
Switch to Camera User 2	Ctrl+2
Switch to Front camera view	F
Switch to Top camera view	T
Switch to First Person view	P
Switch to Listener 0 view	0
Switch to Listener 1 view	1
Switch to Listener 2 view	2

To	Use this shortcut
Switch to Listener 3 view	3
Switch to Listener 4 view	4
Switch to Listener 5 view	5
Switch to Listener 6 view	6
Switch to Listener 7 view	7

Game Profiler

To	Use this shortcut
Open the Profiler Settings dialog box	Alt+G
Navigate up to the next blue circle	Ctrl+Up arrow
Navigate down to the next blue circle	Ctrl+Down arrow
Force time cursor to capture log entry	Shift+Click

Global - Contextual (Active view and selected object)

To	Use this shortcut
For Property Values	
Reset to default value	Ctrl+click
Fine-tune value using the super slider	Shift+drag
Offset values in multiple selection using the super slider.	Alt+drag

Music Segment Editor

To	Press
Jump to the beginning of the segment	Home
Jump to the end of the segment	End
Jump to first track	Shift+Home
Jump to last track	Shift+End
Jump to top left	Ctrl+Home
Jump to bottom right	Ctrl+End
Move play cursor to Entry cue	0
Move play cursor to Exit cue	1
Move play cursor to custom cues	2 through 8
Move play cursor to beginning of segment	.(period)
Snap to bar/beats	Q
Snap to cues	W
Snap to clips/loops	E
Split on cursor	S
Zoom in	+
Zoom in (selection)	Z+marquee selection
Zoom out	-

To	Press
Pan	X+mouse drag
Insert a cue at play cursor	Insert
Set the Entry cue	Ctrl+click ruler
Set the Exit cue	Alt+click ruler
Set new custom cue	Shift+click ruler
Move Entry/Exit cues to selection	P

Also refer to [Zoomable Editor](#).

Position Editor (User-defined)

The shortcuts for the Position Editor (User-defined) are grouped into the following sections:

- [In the Path List](#)

Refer to [Zoomable Editor](#) for the shortcuts in the path editor and the timeline view.

In the Path List

To	Use this shortcut
Navigate Up and Down	Up/Down arrows

Project Explorer

To	Use this shortcut
Expand node	Right arrow
Collapse node	Left arrow
Navigate up	Up arrow
Navigate down	Down arrow

RTPC Graph View

Refer to [Zoomable Editor](#).

Schematic View

To	Use this shortcut
Change selection, navigate the project, expand or collapse the hierarchy	Arrows
Go to the first child of a group of sibling objects	Home
Go to the last child of a group of sibling objects	End
Go to the Master Audio or Master Motion Bus	Ctrl+Home

To	Use this shortcut
Edit selection in its Property Editor	Enter
Playback	Space
Go to the Search field	Ctrl+F
Go to the project hierarchy (from the Search field)	Enter

Soundcaster

To	Use this shortcut
Stop all	Shift+Spacebar
Pause all/Resume all	Ctrl+Shift+Spacebar
Toggle Original mode	Alt+O
Toggle PF Only mode	Alt+I
Reset all	Ctrl+Alt+R

Transport Control

To	Use this shortcut
Play/Stop	Spacebar
Pause/Resume	Ctrl+Spacebar
Toggle Original mode	Alt+O
Toggle PF Only mode	Alt+I
Pin/Unpin	Ctrl+Alt+P
Load and pin in Transport Control	Alt+click
Toggle to State	Alt+S
Toggle to Switch	Alt+W
Toggle to RTPC	Alt+R
Toggle to Trigger	Alt+T
Reset All	Ctrl+Alt+R

Glossary

AAC	A perceptual coding audio compression method that can be used in Wwise on the Mac and iOS platforms. AAC is said to achieve better sound quality than MP3 at similar bit rates. Compression is variable, content dependent, and the quality setting can be controlled by the "quality" slider. On iOS, AAC is decoded by the hardware assisted codec if it is available. Note that the iOS hardware may only decode one AAC sound at a time.
Absolute Property	Properties, such as positioning and playback priority, that are usually defined at the top-level parent object and automatically passed down to each of the parent's child objects. You can override the top-level parent properties by defining these properties at a different level within the hierarchy.
Actor-Mixer	A hierarchical structure of one or more sounds, motion objects, containers, and/or actor-mixers. You can use an actor-mixer to control properties for all objects below it.
ADPCM	Adaptive Differential Pulse Code Modulation (ADPCM) is an audio file conversion encoding method that quantizes the difference between a sound signal and a prediction that has been made of that sound signal. The ADPCM quantization step is adaptive, which results in better audio quality for the compressed signals. This method differs from PCM encoding where the signals are quantized directly.
ATRAC9	A perceptual encoding method for the PS Vita and PS4 that permits encoding of audio files at various bit rates while maintaining a very good perceived sound quality.
Attenuation	The decrease in volume of a sound, music, or motion FX object, as it moves away from the source emitter.
Audio Input	The Audio Input is a sample source plug-in that allows audio content generated by the game to be sent through the Wwise pipeline and processed by the sound engine.
Audio Source	A separate abstraction layer between the audio file and the sound object. It is linked to the audio file imported into your project. The audio source remains linked to the audio file that you imported into your project so that you can reference it at any time.
Auto-ducking	The action of lowering the volume level of one audio signal in order for another simultaneous audio signal to have more prominence.

Auxiliary Bus	A special type of audio bus that is generally used to apply effects such as Reverbs and Delays for simulating environmental effects or to allow dynamic mixing (Side-Chaining).
Auxiliary Send	An audio signal routing technique used to send an audio signal to an auxiliary bus. An Auxiliary Send can be controlled either per sound object, or per game object when using the SDK API.
Bit Depth	The number of bits used to describe each sample within a digital audio file. In PCM audio, the bit depth determines the maximum possible dynamic range of the signal.
Bit Rate	The amount of data, specifically bits, transmitted or received per second. The higher the bit rate, the more file data is processed and usually the higher the resolution.
Blend Container	A group of one or more sounds, motion objects, and/or containers that are played back simultaneously. The objects within this container can be grouped into blend tracks where properties are mapped to game parameter values using RTPCs. Crossfades can also be applied between the objects within a blend track based on the value of a game parameter.
Cache	A project folder that contains all the converted files for the platforms that you are developing. By default, this folder is stored locally although you can modify its location. Multiple users should not access the cache folder simultaneously.
Cent	A unit of pitch equal to 1/100 of a semitone. An octave consists of 1200 cents.
Child Object	An object within a hierarchical structure that lies within a higher level or parent object.
Clip	A music object that represents an audio source. Clips are arranged in music tracks.
Compressor	An audio effect plug-in that reduces the dynamic range of a signal by weakening any part of the input signal that is above a pre-defined threshold value.
Container	A group of one or more objects, including sounds, motion FX objects, and/or containers that are played according to a certain defined behavior. In Wwise, there are several different types of containers, including random, sequence, switch, blend, music switch, and music playlist containers.

Conversion Settings	A group of audio file parameters, which includes sample rate, audio format, and number of channels, that defines the overall quality, memory and CPU usage of the audio files for each platform.
Convolution Reverb	An audio effect that uses IR (Impulse Responses) to simulate the acoustics of real spaces, such as a concert halls, buildings, streets, vehicle interiors, rooms, fields, forests, and others.
Cue	A marker appended to music segments to indicate a key point, such as its entry or exit point.
Custom Cue	A user-created marker appended to music segments to indicate a key sync point. The Entry and Exit cues are not considered custom cues.
dB Scaling	An option to display the graph view Y axis in logarithmic scaling when representing properties that are measured in decibels.
dBFS	(Decibels full scale) Decibel amplitude levels in digital systems which have a maximum available level (like PCM encoding).
DC Offset	The Direct Current represents the center of a waveform of a sound. The offset represents the percentage away from the 0.0 point that the center of a waveform lies.
Default Work Unit	XML files that contain all the information in your project related to the specific element for which they were created. For example, the Default Work Unit.wwu file for Events contains all the information related to events, and the Default Work Unit.wwu file for States contains all the information related to states, and so on. Default work units are created when the project is created.
Definition File	Text files that list all the events in a game, classified by SoundBank.
Delay	An audio effect plug-in that adds echoes by delaying an audio signal for a specified period of time.
Depot	A central repository of files on the Perforce source control server. It contains all versions of all files ever submitted to the server.
Dialogue Event	A method to trigger audio in game using a set of rules or conditions expressed in arguments and argument values that match the possible conditions in the game. These argument values are arranged into argument paths and then assigned to an object in Wwise. When a dialogue event is called by the game, the game matches its current situation with those defined in the dialogue event and the appropriate piece of dialogue is played.

Dithering	The noise added to a signal prior to quantization which reduces the distortion and noise modulation resulting from the quantization process. Although there is a slight increase in the noise level, spectrally shaped dither can minimize the apparent increase. The noise is less objectionable than the distortion, and allows low-level signals to be heard more clearly.
Downmix	A process whereby all channels within a multi-channel source are mixed into a compatible version with fewer channels, such as stereo or mono.
Dry Signal	Output that consists entirely of the original, unprocessed signal.
Echo Density	The amount of echoes per second produced by the reverberation algorithm.
Effect Chain	A series of effects that have been applied to an object or bus in a specific order.
Effect Instance	A customized set of effect properties that can be saved and applied to other objects or busses. Effect instance properties can also be shared across objects.
Empty Event	An event that doesn't contain any actions or objects.
Environmental Effect	An effect that alters the property set of the sounds generated by a game object depending on the position of that object in a game's geometry.
Event (Action Event)	A method to trigger audio in game using an action or series of actions, such as play, mute, and pause, that have been applied to one or more Wwise objects.
Expander	The Wwise Expander plug-in effect expands the dynamic range of a signal by weakening any part of the input signal that is below a pre-defined threshold value. When the signal is soft and below the threshold, the Expander begins to reduce the signal's gain. When the signal is at or louder than the threshold value, no gain reduction is applied to the signal.
External Source	A source plug-in that associates a sound object with an audio file at runtime. It allows the management of large amounts of dialogue lines that could otherwise require a great deal of overhead. It helps to save some runtime memory and it simplifies the replacement of audio files that is sometime required when generating DLC content for a game.

File Manager	A dialog box that displays information about project files and original imported source files, as well as manages many source control plug-in functions, where applicable.
Flanger	An audio effect that mixes two identical signals together, where one of the signals is time-delayed by a small and gradually changing amount producing a swept comb filter effect.
Flat View	A view that is docked into a layout.
Folder	High level structure in the Actor-Mixer hierarchy, used to manage other structures of actor-mixers, containers, and so on.
Frequency	The number of cycles per unit of time.
Gain	A change in the power or amplitude of a signal.
Game Object	Entity in a game to which elements such as an interface, a trigger, or a sound can be attached.
Game Parameter	A parameter from the game, such as speed and RPMs in a car racing game for example, that can be mapped to Wwise property values using RTPCs.
Game Sync	A group of Wwise elements that includes States, Switches, RTPCs, Triggers, and Arguments which are called by conditions in the game and modify the audio and motion accordingly.
Game Unit	The basic unit of length used in calculating game geometry. For example, a stealth FPS might use meters as a game unit, while a space conquest game might use light-years.
Guitar Distortion	An audio effect that alters the shape of the audio waveform and introduces frequency components not present in the original signal. The Wwise Guitar Distortion effect mimics the behavior of commonly used distortion 'stomp boxes' to obtain typical guitar distortion sounds.
Harmonics	Multiples of the fundamental frequency. For example, if the frequency is 50 Hz, the second harmonic is 100 Hz, the third harmonic is 150 Hz, and so on.
Harmonizer	An audio effect that adds one or two pitched voices to the incoming signal.
Headroom	The level difference (in dB) between normal operating level and clipping level in an amplifier or audio device.

High-Pass Filter	A recursive filter that attenuates frequencies lower than the cut-off frequency. The units for this filter represent the percentage of high pass filtering effect that has been applied, where 0 means no high pass filtering (signal unaffected) and 100 means maximal attenuation.
HRTF	Head Related Transfer Function is the way in which sound is received through different physical media from a point in space to the ear. Paired HRTFs allow the synthesis of a binaural sound that seems to originate at a particular point in space.
Imported folder	The hidden project .cache folder containing copies of audio files imported into the project that have undergone a special import conversion process.
Indicator	A specific icon in the Wwise interface that indicates the status of a particular property value. For example, the link indicator shows whether a property value is linked across platforms.
Inharmonicity	A fractional offset between the pitch of a partial and the pitch of the true harmonic (a whole number multiple of the fundamental frequency).
Initialization Bank	A special type of bank that contains all the general information of a project, including information on the bus hierarchy, and information on states, switches, and RTPCs. There is only one Initialization bank per project and it is named "Init.bnk" by default. The Initialization bank must be the first bank loaded when starting a game. If it is not loaded first, the SoundBanks may refuse to load.
Input Range	The complete range of values that can be entered for a property - as opposed to Slider Range.
Integrity Report	Report generated in Wwise that displays errors or issues in a project along with suggested fixes.
Interactive Music	A music composition and arrangement method to create musical scores that are both modular and responsive to in-game actions.
Invalid Events	Events that have been deleted from your project, but are still included in a SoundBank.
IR (Impulse Response)	An audio file resulting from the measurement of real acoustic characteristics of a location, such as a concert hall. Impulse Responses are used in the Convolution Reverb effect to enable the

	acoustic characteristics of a particular location to be applied to the incoming signal.
Latency	Time delays inherent in internal processing or generation of audio signals within a computer.
Layout	A series of views grouped together to facilitate the work involved for a particular task or job.
LFE	Low Frequency Effect. The name of the audio channel specifically intended for deep, low-pitched sounds ranging from 10-120 Hz. The LFE channel is a totally independent channel that must be imported into Wwise as a x.1 media file.
Limiting	An extreme form of compression where the input/output relationship becomes very flat (10:1 or higher). This places a hard limit on the signal level.
Listener	A virtual microphone or motion sensor in the game that helps assign the sounds to particular speakers or motion to particular motors to simulate a 3D environment.
Look-ahead time	In streaming, this refers to the time reserved for the sound engine to seek the streaming data.
Low-Pass Filter	A recursive filter that attenuates frequencies higher than the cut-off frequency. The units for this filter represent the percentage of low pass filtering effect that has been applied, where 0 means no low pass filtering (signal unaffected) and 100 means maximal attenuation.
Master-Mixer Hierarchy	A bus or series of busses at the top of your project hierarchy that allow you to re-group many different sound, music, and motion structures according to the main categories within a game. For example, you can group all the music structures under one audio bus, all the sound effects under another audio bus, and so on.
Matrix Reverb	A unique reverb effect optimized for game production that balances quality with performance and includes real-time editing and RTPC mapping functionalities.
Meter Effect	An audio effect that measures the level of a signal without modifying it, and optionally outputs this level as a Game Parameter. It is most useful for achieving side-chaining, where the measured level of a bus drives the volume of another bus through RTPC.

Meter (Peak)	A series of meters that display the level of the audio signal for each channel. While audio and auxiliary busses show the output signal, dynamic effects (i.e. compressors and limiters) generally display the audio input, audio output and the applied gain reduction.
Mixing Desk	A flexible and powerful mixing console that regroups a variety of bus and object properties into a single view, used to fine-tune the audio mix of your game in real-time.
Mixing Session	A set of Wwise objects of your choice used within the Mixing Desk that can be saved and re-used at any time.
Modal density	Modes are the peaks in the frequency domain representation of an audio signal. Increasing the modal density improves the realism of the reverberation when simulating most acoustic spaces. Decreased modal density can cause ringing sounds.
Motion FX Object	A representation in Wwise of the individual motion assets that you have created for your project. Each motion FX object contains one or more sources, which defines the actual motion that will be generated in-game.
Music Clip	The basic component of a music track displayed as a rectangular area representing a single WAV file.
Music Playlist Container	A group of one or more music objects and/or containers that are played back in a random or sequential order.
Music Segment	A multi-track music object that is the basic unit of the Interactive Music hierarchy.
Music Switch Container	A group of one or more music objects and/or containers that are played back according to the switch or state that is called.
Music Track	A music object that contains arrangements of individual music clips that are displayed in waveform so that you can visually align them in a music segment.
Nested Object	An object that lies within another object.
Nested Work Units	A work unit nested inside another work unit. It allows for a finer granularity in the project files, reducing potential conflicts in a workgroup environment when merging files under source control.
Noise Gate	An effect created in the Expander effect plug-in that removes sounds from the output signal almost completely. A noise gate is

	created by setting a high expander ratio (over 10:1) that closes the gate for sounds whose gain has been reduced to this extent.
Nyquist Frequency	The highest audio frequency that can be accurately sampled, equivalent to one-half of the sampling rate. The Nyquist sampling theorem showed that the sampling rate must be at least twice the highest frequency present in the sample in order to accurately reconstruct the original signal.
Object	Elements in Wwise, such as the Sounds, Motion FX, Containers, and Actor-Mixers, that are used to contain, group, and define sounds, voices, motion, and music within the project hierarchy.
Obstruction	A condition that occurs when an object in the game, such as a pillar, partially blocks the space between a sound object and a listener.
Occlusion	A condition that occurs when an object in the game, such as a wall, completely blocks the space between a sound object and a listener.
Originals folder	The folder containing untouched copies of the audio files imported into the project. This folder is usually stored under source control.
Orphan file	Audio files no longer associated with a sound, motion FX, or music object. These files are not automatically deleted when you delete a sound object. To delete these files, clear the audio .cache folder.
Output buffer latency	Latency introduced during audio playback determined by the number of output buffers used by Wwise.
Parametric EQ	An audio effect plug-in that allows you to apply a variety of filters to shape the spectrum of your sound.
Parent Object	An object within a hierarchical structure that contains child objects.
Passband	The band of frequencies that passes through a filter with essentially no attenuation.
PCM	Pulse code modulation. A method for encoding audio files where distinct binary representations or pulse codes are chosen, and these are quantized by measuring values between two encoded points, selecting the value associated with the nearest point.
PCM frame	A PCM frame is composed of samples for all channels at a given time. Each frame represents 1/sample frequency seconds.

Peak Limiter	An audio effect that controls the dynamic range of audio signals. It does this by weakening parts of the audio signal that briefly exceed a pre-defined threshold value as calculated with peak-based detection.
Physical Folder	A directory on your hard disk, under your Wwise project root, that can contain other physical folders or work units used in your project. Physical folders cannot be child objects for containers, motion FX, or sounds.
Physical Voice	The physical environment of the game where audio and motion are played and processed by the sound engine. When volume levels become very low, sounds and motion objects can move into a virtual environment where they are managed and monitored by the sound engine, but no audio processing is performed.
Pitch	The playback speed of the sound or motion object.
Pitch Shifter	An audio effect that changes the pitch without affecting the duration of the resulting audio signal.
Point Source	A source that emits sound or motion as if from a single point.
Post-exit	The segment area after the exit cue that can be used for transitions in interactive music.
Pre-entry	The segment area before the entry cue that can be used for transitions in interactive music.
Prefetch Time	In streaming, this refers to a small buffer that covers the latency time required to fetch the rest of the file data.
Preset	A customized set of properties for objects, effects, and sound/motion propagation that can be saved and re-used at any time.
Quantization	The process whereby a range of values of an audio file are divided into sub ranges, each of which is represented by an assigned value.
Query	A specific set of search criteria used to find a particular object or project element.
Random Container	A group of one or more sounds, motion FX objects, and/or containers that are played back in a random order.
Randomizer	An special effect in Wwise that is applied to a property value that allows you to define a range of possible values that can be used randomly each time an object is played.

Recursive Filter	A filter that uses previously-calculated output values in addition to the current input values to calculate the latest output. Also known as feedback filters.
Relative Property	Properties, such as volume and pitch, that can be defined at each level within your project hierarchy. These property values are cumulative, which means that a parent's property values are added to those of the child.
Reverb	An audio effect plug-in that simulates the acoustics of a particular room or space.
Ripple (in passband)	The difference between the maximum and minimum attenuation within a passband.
RMS power	Root Mean Square. This is a measure of a signal's average amplitude that provides in most cases a better approximation of a signal's power than using peak amplitude. This value is obtained by summing the square of the samples over a given time window and taking the square root of the result.
RoomVerb	A versatile, high-quality reverb effect plug-in that simulates the acoustics of a particular room or space.
RTPC	Real Time Parameter Controls. An interactive method used to drive the audio or motion in game by mapping game parameter values to properties in Wwise. RTPCs can also be used to drive switch changes by mapping game parameters to switch groups.
Sample Rate	The frequency at which an audio signal is sampled per second during conversion to a digital signal or during a digital conversion.
Send Volume	The level or amplitude of the audio signal sent to an auxiliary bus.
Sequence Container	A group of one or more sounds, motion FX objects, and/or containers that are played back according to a specific playlist.
ShareSet	A set of properties that can be shared between objects to define attributes such as effects or attenuation.
Silence Source	A plug-in source of a specified duration that contains no sound or motion.
Side-Chaining	Monitoring the level of an audio signal and using the audio information to manipulate another audio signal in real-time. This technique is used to give more emphasis to the important sounds

	in the final mix by automatically controlling the volume of the less important sounds.
Slider Range	The default range of values that can be entered for a property using the slider - as opposed to Slider Range.
SoundBank	A group of event data, sound, music, and motion structure data, and/or media files that will be loaded into the game's platform memory at a particular point in a game.
Soundcaster	The view in Wwise that provides transport controls, auditioning, and real-time mixing in game or in Wwise, for objects or events that can be inserted and removed as needed.
Soundcaster Session	A saved project element that contains the Wwise objects and events used in a specific simulation created in the Soundcaster.
SoundFrame	A unique plug-in interface that enables external applications to seamlessly communicate with Wwise. Examples of some of the tools that can be built with SoundFrame are SFTest and CarSim.
Sound Object	A representation in Wwise of the individual audio assets that you have created for your project. Each sound object contains one or more sources, which defines the actual audio content that will be played in-game.
Sound SFX	A sound object within the Actor-Mixer Hierarchy that contains sounds, music, and ambiences.
Sound Voice	A sound object that contains game voice over or dialogue.
Source Control	The management of changes to documents (i.e. Wwise Project , Work Units, Audio files, etc.) where each revision of the document is tagged with a revision identification number, and associated with a timestamp and the name of the person who modified the document. In most version control system, revisions can be compared, restored, and merged.
Source Plug-in	An audio or motion source that is created by a plug-in coming from outside of Wwise.
Spatialization	A function in Wwise that determines the actual location or position of the sound or music object within the 3D environment of the game.
State	A global offset or adjustment to the game audio or motion properties that relates to changes in the physical and environmental conditions in the game.

Stereo Delay	An audio effect that provides a dual channel delay with a built-in filter. It has feedback and crossfeed controls to send delayed signals from one channel to another to create stereo effects.
Stinger	A brief musical phrase that is superimposed and mixed over the currently playing music.
Switch	A switch represents the alternatives that exist for a particular element in game, and is used to help manage the corresponding sounds or motion objects for these alternatives. For example, if a character is running on a concrete surface and then moves onto grass, the sound of the footsteps in a switch container should change to match the change of surface.
Switch Container	A series of switches or states, each of which contains a group of sounds, motion FX objects, or containers, that correspond to particular changes in the environment of the game. For example, a switch container for a character's footsteps might contain switches for grass, concrete, wood and any other surface that a character can walk on in the game.
Tab-delimited File	A special kind of plain text file where the information is arranged into columns separated by tabs.
Time Stretch	An audio effect that changes the duration without affecting the pitch of the resulting audio signal.
Transition	In Interactive Music, a transition is meant to be a smooth bridge between a source and destination music segment.
Transition Time	Time period used to transition from one state to another within the same state group. During the transition, an interpolation of the two state properties occurs.
Tremolo	An audio effect that modulates the amplitude of the input signal with a unipolar carrier signal.
Trigger	A game sync that responds to a spontaneous occurrence in the game and launches a stinger.
VAG	An audio file conversion encoding method for the PS Vita based on ADPCM encoding.
Virtual Folder	An organizational object, displayed as a folder and contained within a work unit or one of its child objects, in which you can place other objects, such as virtual folders, actor-mixers, containers, motion FX, and sounds. Virtual folders cannot be child objects for

	containers, motion FX, or sounds and do not have a corresponding directory on your hard disk.
Virtual Voice	A virtual environment where sounds and motion are managed and monitored by the sound engine, but no processing is performed. Objects move into the virtual voice when their volume levels fall below the volume threshold.
Voice	A separate or discrete playback instance, either audio or motion.
Voice Starvation	A type of error message that is displayed in the Capture Log when the sound engine cannot provide audio data to the platform hardware buffer in a timely manner. This type of problem occurs when there is excessive use of the host CPU. For example, when the platform CPU is trying to mix too many sources or is using too many audio effects simultaneously.
Volume	The amplitude or level of intensity of the audio or motion output.
Volume Threshold	A certain volume level below which the behavior of sound, music, and motion objects can be specifically determined. For example, voices that fall below the volume threshold can either continue to play, be killed, or sent to the virtual voice list. These behaviors are defined on the Advanced Settings tab of the object's Property Editor.
Vorbis	A perceptual encoding method that permits encoding of audio files at various bit rates while maintaining a very good perceived sound quality. The balance between data compression efficiency and perceived sound quality is controlled using the Quality Factor setting or by specifying the maximum, minimum, and average bit rates per channel.
Watch	An individual process you can use to monitor the game objects and listeners in your game.
WAVEFORMATEXTENSIBLE	A structure that defines the format of waveform-audio data for formats having more than two channels. To preserve the specific multi-channel configuration of a WAV file on input, you must define the channel information in the WAV header file as part of the channel mask of the WAVEFORMATEXTENSIBLE.
Wet Signal	Output that consists entirely of processed sound.
Workgroup	A group of people working concurrently on the same Wwise project.

Work Unit	A distinct XML file that contains information related to a particular section or element within your project.
World Builder	An application used to create the virtual environment of your game.
XMA	A hardware supported perceptual coding audio compression method for the Xbox 360. XMA is a console-optimized version of Windows Media Audio Pro. Compression is variable, content dependent, and the quality setting can be controlled by the "Compression Quality" slider. The most recent version, XMA 2.0, contains a new block size parameter for creating a seek table that facilitates seeking XMA data.
xWMA	A software supported perceptual coding audio compression method for the Xbox 360. xWMA is a console-optimized version of Windows Media Audio Pro, and provides a greater compression ratio than XMA. Compression is variable, content dependent, and the quality setting can be controlled by the "bit rate" drop-down menu. Note that xWMA has restrictions on looping.