

OSSC

0.0.4

Generated by Doxygen 1.8.13



# Contents

<b>1</b>	<b>Namespace Documentation</b>	<b>1</b>
1.1	OSSC Namespace Reference . . . . .	1
1.2	OSSC.Editor Namespace Reference . . . . .	2
1.3	OSSC.Model Namespace Reference . . . . .	2
<b>2</b>	<b>Class Documentation</b>	<b>3</b>
2.1	OSSC.Model.CategoryItem Class Reference . . . . .	3
2.1.1	Detailed Description . . . . .	3
2.1.2	Member Data Documentation . . . . .	3
2.1.2.1	audioObjectPrefab . . . . .	4
2.1.2.2	categoryVolume . . . . .	4
2.1.2.3	foldOutSoundItems . . . . .	4
2.1.2.4	isMute . . . . .	4
2.1.2.5	name . . . . .	4
2.1.2.6	soundItems . . . . .	5
2.1.2.7	soundsSearchName . . . . .	5
2.1.2.8	usingDefaultPrefab . . . . .	5
2.2	OSSC.CueManager Class Reference . . . . .	5
2.2.1	Detailed Description . . . . .	6
2.2.2	Constructor & Destructor Documentation . . . . .	6
2.2.2.1	CueManager() [1/2] . . . . .	6
2.2.2.2	CueManager() [2/2] . . . . .	6
2.2.3	Member Function Documentation . . . . .	6
2.2.3.1	GetSoundCue() . . . . .	6

2.2.3.2	StopAllCues()	7
2.3	OSSC.Model.CustomRange Class Reference	7
2.3.1	Detailed Description	7
2.3.2	Member Function Documentation	7
2.3.2.1	GetRandomRange()	8
2.3.3	Member Data Documentation	8
2.3.3.1	max	8
2.3.3.2	min	8
2.4	IPoolable Interface Reference	8
2.4.1	Detailed Description	9
2.4.2	Member Function Documentation	9
2.4.2.1	IsFree()	9
2.4.3	Property Documentation	9
2.4.3.1	pool	9
2.5	OSSC.ISoundCue Interface Reference	10
2.5.1	Detailed Description	10
2.5.2	Member Function Documentation	10
2.5.2.1	Pause()	11
2.5.2.2	Play()	11
2.5.2.3	Resume()	11
2.5.2.4	Stop()	11
2.5.3	Property Documentation	12
2.5.3.1	AudioObject	12
2.5.3.2	Data	12
2.5.3.3	ID	12
2.5.3.4	IsPlaying	12
2.5.3.5	OnPlayCueEnded	12
2.5.3.6	OnPlayEnded	13
2.6	ObjectPool Class Reference	13
2.6.1	Detailed Description	13

2.6.2	Member Function Documentation	13
2.6.2.1	GetFreeObject()	13
2.6.3	Member Data Documentation	14
2.6.3.1	pools	14
2.7	OSSC.PlaySoundSettings Struct Reference	14
2.7.1	Detailed Description	15
2.7.2	Member Function Documentation	15
2.7.2.1	Init()	15
2.7.3	Member Data Documentation	15
2.7.3.1	categoryName	16
2.7.3.2	fadeInTime	16
2.7.3.3	fadeOutTime	16
2.7.3.4	isLooped	16
2.7.3.5	name	16
2.7.3.6	names	17
2.7.3.7	parent	17
2.7.3.8	soundCueProxy	17
2.7.3.9	tagName	17
2.8	PrefabBasedPool Class Reference	17
2.8.1	Detailed Description	18
2.8.2	Constructor & Destructor Documentation	18
2.8.2.1	PrefabBasedPool()	18
2.8.3	Member Function Documentation	18
2.8.3.1	Despawn()	18
2.8.3.2	GetFreeObject()	18
2.8.4	Member Data Documentation	18
2.8.4.1	parent	19
2.8.4.2	pool	19
2.8.4.3	prefab	19
2.9	OSSC.SoundController Class Reference	19

2.9.1	Detailed Description	20
2.9.2	Member Function Documentation	20
2.9.2.1	Play()	20
2.9.2.2	SetMute()	20
2.9.2.3	StopAll()	21
2.9.3	Member Data Documentation	21
2.9.3.1	_database	21
2.9.3.2	_defaultPrefab	21
2.9.4	Property Documentation	21
2.9.4.1	defaultPrefab	22
2.10	OSSC.Model.SoundControllerData Class Reference	22
2.10.1	Detailed Description	22
2.10.2	Member Data Documentation	22
2.10.2.1	assetName	23
2.10.2.2	foldOutCategories	23
2.10.2.3	foldOutTags	23
2.10.2.4	items	23
2.10.2.5	soundTags	23
2.11	OSSC.Editor.SoundControllerEditor Class Reference	24
2.11.1	Detailed Description	24
2.11.2	Member Function Documentation	24
2.11.2.1	OnInspectorGUI()	24
2.12	OSSC.SoundCue Class Reference	24
2.12.1	Detailed Description	25
2.12.2	Constructor & Destructor Documentation	25
2.12.2.1	SoundCue() [1/2]	26
2.12.2.2	SoundCue() [2/2]	26
2.12.3	Member Function Documentation	26
2.12.3.1	Pause()	26
2.12.3.2	Play() [1/2]	26

2.12.3.3	<a href="#">Play() [2/2]</a>	27
2.12.3.4	<a href="#">Resume()</a>	28
2.12.3.5	<a href="#">Stop()</a>	28
2.12.4	<a href="#">Property Documentation</a>	28
2.12.4.1	<a href="#">AudioObject</a>	28
2.12.4.2	<a href="#">Data</a>	29
2.12.4.3	<a href="#">ID</a>	29
2.12.4.4	<a href="#">IsPlaying</a>	29
2.12.4.5	<a href="#">OnPlayCueEnded</a>	29
2.12.4.6	<a href="#">OnPlayEnded</a>	29
2.12.4.7	<a href="#">OnPlayKilled</a>	30
2.13	<a href="#">OSSC.SoundCueData Struct Reference</a>	30
2.13.1	<a href="#">Detailed Description</a>	30
2.13.2	<a href="#">Member Data Documentation</a>	30
2.13.2.1	<a href="#">audioPrefab</a>	31
2.13.2.2	<a href="#">categoriesForSounds</a>	31
2.13.2.3	<a href="#">categoryVolumes</a>	31
2.13.2.4	<a href="#">fadeInTime</a>	31
2.13.2.5	<a href="#">fadeOutTime</a>	31
2.13.2.6	<a href="#">isFadeIn</a>	32
2.13.2.7	<a href="#">isFadeOut</a>	32
2.13.2.8	<a href="#">isLooped</a>	32
2.13.2.9	<a href="#">sounds</a>	32
2.14	<a href="#">OSSC.SoundCueProxy Class Reference</a>	32
2.14.1	<a href="#">Detailed Description</a>	33
2.14.2	<a href="#">Member Function Documentation</a>	33
2.14.2.1	<a href="#">Pause()</a>	33
2.14.2.2	<a href="#">Play()</a>	34
2.14.2.3	<a href="#">Resume()</a>	34
2.14.2.4	<a href="#">Stop()</a>	34

2.14.3	Property Documentation	34
2.14.3.1	AudioObject	34
2.14.3.2	Data	35
2.14.3.3	ID	35
2.14.3.4	IsPlaying	35
2.14.3.5	OnPlayCueEnded	35
2.14.3.6	OnPlayEnded	35
2.14.3.7	SoundCue	36
2.15	OSSC.Model.SoundItem Class Reference	36
2.15.1	Detailed Description	36
2.15.2	Member Data Documentation	36
2.15.2.1	clips	37
2.15.2.2	isRandomPitch	37
2.15.2.3	isRandomVolume	37
2.15.2.4	mixer	37
2.15.2.5	name	37
2.15.2.6	pitchRange	38
2.15.2.7	tagID	38
2.15.2.8	volume	38
2.15.2.9	volumeRange	38
2.16	OSSC.SoundObject Class Reference	38
2.16.1	Detailed Description	39
2.16.2	Member Function Documentation	39
2.16.2.1	IsFree()	40
2.16.2.2	Pause()	40
2.16.2.3	Play()	40
2.16.2.4	Resume()	40
2.16.2.5	Setup()	40
2.16.2.6	Stop()	41
2.16.3	Member Data Documentation	41



2.16.3.1	OnFinishedPlaying	41
2.16.4	Property Documentation	41
2.16.4.1	clipName	41
2.16.4.2	ID	42
2.16.4.3	isDespawnOnFinishedPlaying	42
2.16.4.4	source	42
2.17	OSSC.Editor.SoundObjectEditor Class Reference	42
2.17.1	Detailed Description	43
2.17.2	Member Function Documentation	43
2.17.2.1	OnInspectorGUI()	43
2.18	OSSC.SoundTags Class Reference	43
2.18.1	Detailed Description	44
2.18.2	Constructor & Destructor Documentation	44
2.18.2.1	SoundTags()	44
2.18.3	Member Function Documentation	44
2.18.3.1	GetTagDataByID()	44
2.18.3.2	GetTagDataByName()	44
2.18.3.3	GetTagIDByName()	45
2.18.3.4	GetTagNameByID()	45
2.18.3.5	RemoveByTag()	46
2.18.3.6	SetTag()	46
2.18.3.7	ToArray()	46
2.18.3.8	ToArrayIDs()	46
2.18.3.9	ToArrayNames()	47
2.19	OSSC.TagData Struct Reference	47
2.19.1	Detailed Description	47
2.19.2	Member Data Documentation	47
2.19.2.1	ID	47
2.19.2.2	name	47

<b>3</b>	<b>File Documentation</b>	<b>49</b>
3.1	F:/git-projects/audio-controller-unity/AudioController/Assets/OSSC/Source/CueManager.cs File Reference	49
3.2	CueManager.cs	49
3.3	F:/git-projects/audio-controller-unity/AudioController/Assets/OSSC/Source/Editor/SoundController↔ Editor.cs File Reference	50
3.4	SoundControllerEditor.cs	50
3.5	F:/git-projects/audio-controller-unity/AudioController/Assets/OSSC/Source/Editor/SoundObject↔ Editor.cs File Reference	54
3.6	SoundObjectEditor.cs	55
3.7	F:/git-projects/audio-controller-unity/AudioController/Assets/OSSC/Source/IPoolable.cs File Reference	55
3.8	IPoolable.cs	55
3.9	F:/git-projects/audio-controller-unity/AudioController/Assets/OSSC/Source/ISoundCue.cs File Reference	56
3.10	ISoundCue.cs	56
3.11	F:/git-projects/audio-controller-unity/AudioController/Assets/OSSC/Source/Model/CategoryItem.cs File Reference	56
3.12	CategoryItem.cs	57
3.13	F:/git-projects/audio-controller-unity/AudioController/Assets/OSSC/Source/Model/SoundController↔ Data.cs File Reference	57
3.14	SoundControllerData.cs	57
3.15	F:/git-projects/audio-controller-unity/AudioController/Assets/OSSC/Source/Model/SoundItem.cs File Reference	58
3.16	SoundItem.cs	58
3.17	F:/git-projects/audio-controller-unity/AudioController/Assets/OSSC/Source/ObjectPool.cs File Reference	58
3.18	ObjectPool.cs	59
3.19	F:/git-projects/audio-controller-unity/AudioController/Assets/OSSC/Source/SoundController.cs File Reference	60
3.20	SoundController.cs	60
3.21	F:/git-projects/audio-controller-unity/AudioController/Assets/OSSC/Source/SoundCue.cs File Reference	64
3.22	SoundCue.cs	64
3.23	F:/git-projects/audio-controller-unity/AudioController/Assets/OSSC/Source/SoundCueProxy.cs File Reference	67
3.24	SoundCueProxy.cs	67
3.25	F:/git-projects/audio-controller-unity/AudioController/Assets/OSSC/Source/SoundObject.cs File Reference	68
3.26	SoundObject.cs	69
3.27	F:/git-projects/audio-controller-unity/AudioController/Assets/OSSC/Source/SoundTags.cs File Reference	71
3.28	SoundTags.cs	71
	<b>Index</b>	<b>75</b>

# Chapter 1

## Namespace Documentation

### 1.1 OSSC Namespace Reference

#### Namespaces

- namespace [Editor](#)
- namespace [Model](#)

#### Classes

- class [CueManager](#)  
*Manages all SoundCues*
- interface [ISoundCue](#)  
*[SoundCue](#) Interface. [SoundController](#) returns a [SoundCue](#) Interface to further control the playing SoundCue*
- struct [PlaySoundSettings](#)  
*Set the settings to play a particular cue with particular preferences.*
- class [SoundController](#)  
*The main class that is used for Playing and controlling all sounds.*
- class [SoundCue](#)  
*Plays a whole cue of soundItems*
- struct [SoundCueData](#)  
*Used for sending data to play to AudioCue*
- class [SoundCueProxy](#)  
*Given by the [SoundController](#) to User as [ISoundCue](#) to control the playing [SoundCue](#).*
- class [SoundObject](#)  
*Used by the [SoundCue](#). Controls the AudioSource.*
- class [SoundTags](#)  
*Used By the [SoundController](#) for tagging SoundItems.*
- struct [TagData](#)  
*Used by the [SoundTags](#) to save Tags.*

## 1.2 OSSC.Editor Namespace Reference

### Classes

- class [SoundControllerEditor](#)  
*Draws the Custom [Editor](#) for [SoundController](#)*
- class [SoundObjectEditor](#)  
*Draw the custom editor inspector for [SoundObject](#)*

## 1.3 OSSC.Model Namespace Reference

### Classes

- class [CategoryItem](#)  
*Used by the [SoundControllerData](#) to store categories.*
- class [CustomRange](#)  
*Used by [SoundItem](#) to store Random Ranges.*
- class [SoundControllerData](#)  
*[SoundController](#)'s Database.*
- class [SoundItem](#)  
*Used by [CategoryItem](#) to store sounds data.*

## Chapter 2

# Class Documentation

### 2.1 OSSC.Model.CategoryItem Class Reference

Used by the [SoundControllerData](#) to store categories.

#### Public Attributes

- string [name](#)  
*Category name*
- [SoundItem](#) [] [soundItems](#)  
*Array of SoundItems*
- GameObject [audioObjectPrefab](#)  
*Alternative [SoundObject](#) prefab to use, instead of the Default one from [SoundController](#).*
- bool [usingDefaultPrefab](#) = true  
*Check whether to use alternative [SoundObject](#) prefab.*
- float [categoryVolume](#) = 1f  
*Volume of the category*
- bool [foldOutSoundItems](#) = false  
*Used for [Editor](#) to save whether the SoundItems are folded out or not.*
- string [soundsSearchName](#) = ""  
*Save the last search name written in editor.*
- bool [isMute](#) = false  
*Is Category mute?*

#### 2.1.1 Detailed Description

Used by the [SoundControllerData](#) to store categories.

Definition at line 11 of file [CategoryItem.cs](#).

#### 2.1.2 Member Data Documentation

#### 2.1.2.1 audioObjectPrefab

```
GameObject OSSC.Model.CategoryItem.audioObjectPrefab
```

Alternative [SoundObject](#) prefab to use, instead of the Default one from [SoundController](#).

Definition at line 24 of file [CategoryItem.cs](#).

#### 2.1.2.2 categoryVolume

```
float OSSC.Model.CategoryItem.categoryVolume = 1f
```

Volume of the category

Definition at line 34 of file [CategoryItem.cs](#).

#### 2.1.2.3 foldOutSoundItems

```
bool OSSC.Model.CategoryItem.foldOutSoundItems = false
```

Used for [Editor](#) to save whether the SoundItems are folded out or not.

Definition at line 39 of file [CategoryItem.cs](#).

#### 2.1.2.4 isMute

```
bool OSSC.Model.CategoryItem.isMute = false
```

Is Category mute?

Definition at line 47 of file [CategoryItem.cs](#).

#### 2.1.2.5 name

```
string OSSC.Model.CategoryItem.name
```

Category name

Definition at line 16 of file [CategoryItem.cs](#).

### 2.1.2.6 soundItems

```
SoundItem [] OSSC.Model.CategoryItem.soundItems
```

Array of SoundItems

Definition at line 20 of file [CategoryItem.cs](#).

### 2.1.2.7 soundsSearchName

```
string OSSC.Model.CategoryItem.soundsSearchName = ""
```

Save the last search name written in editor.

Definition at line 43 of file [CategoryItem.cs](#).

### 2.1.2.8 usingDefaultPrefab

```
bool OSSC.Model.CategoryItem.usingDefaultPrefab = true
```

Check whether to use alternative [SoundObject](#) prefab.

Definition at line 28 of file [CategoryItem.cs](#).

The documentation for this class was generated from the following file:

- [F:/git-projects/audio-controller-unity/AudioController/Assets/OSSC/Source/Model/CategoryItem.cs](#)

## 2.2 OSSC.CueManager Class Reference

Manages all SoundCues

### Public Member Functions

- [CueManager](#) ()  
*Default Constructor*
- [CueManager](#) (int initialSize)  
*Costruct [CueManager](#) with some initial SoundCues created.*
- [SoundCue](#) GetSoundCue ()  
*Get a free [SoundCue](#).*
- void [StopAllCues](#) (bool shouldCallOnEndCallback=true)  
*Stops all SoundCues from playing.*

### 2.2.1 Detailed Description

Manages all SoundCues

Definition at line 10 of file [CueManager.cs](#).

### 2.2.2 Constructor & Destructor Documentation

#### 2.2.2.1 CueManager() [1/2]

```
OSSC.CueManager.CueManager ( )
```

Default Constructor

Definition at line 23 of file [CueManager.cs](#).

#### 2.2.2.2 CueManager() [2/2]

```
OSSC.CueManager.CueManager (
    int initialSize )
```

Costruct [CueManager](#) with some initial SoundCues created.

Parameters

<i>initialSize</i>	Size of the <a href="#">SoundCue</a> pool.
--------------------	--

Definition at line 32 of file [CueManager.cs](#).

### 2.2.3 Member Function Documentation

#### 2.2.3.1 GetSoundCue()

```
SoundCue OSSC.CueManager.GetSoundCue ( )
```

Get a free [SoundCue](#).

Returns

Returns a [SoundCue](#) instance.

Definition at line 41 of file [CueManager.cs](#).



### 2.2.3.2 StopAllCues()

```
void OSSC.CueManager.StopAllCues (
    bool shouldCallOnEndCallback = true )
```

Stops all SoundCues from playing.

#### Parameters

<i>shouldCallOnEndCallback</i>	Check whether to call OnEnd events or not.
--------------------------------	--

Definition at line 52 of file [CueManager.cs](#).

The documentation for this class was generated from the following file:

- F:/git-projects/audio-controller-unity/AudioController/Assets/OSSC/Source/[CueManager.cs](#)

## 2.3 OSSC.Model.CustomRange Class Reference

Used by [SoundItem](#) to store Random Ranges.

### Public Member Functions

- float [GetRandomRange](#) ()  
*Gets a random value from it's Minimum and Maximum limits.*

### Public Attributes

- float [min](#) = 1f  
*Minimum limit*
- float [max](#) = 1f  
*Maximum limit*

### 2.3.1 Detailed Description

Used by [SoundItem](#) to store Random Ranges.

Definition at line 55 of file [SoundItem.cs](#).

### 2.3.2 Member Function Documentation

### 2.3.2.1 GetRandomRange()

```
float OSSC.Model.CustomRange.GetRandomRange ( )
```

Gets a random value from it's Minimum and Maximum limits.

#### Returns

Definition at line 70 of file [SoundItem.cs](#).

## 2.3.3 Member Data Documentation

### 2.3.3.1 max

```
float OSSC.Model.CustomRange.max = 1f
```

Maximum limit

Definition at line 64 of file [SoundItem.cs](#).

### 2.3.3.2 min

```
float OSSC.Model.CustomRange.min = 1f
```

Minimum limit

Definition at line 60 of file [SoundItem.cs](#).

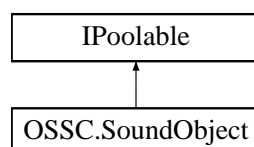
The documentation for this class was generated from the following file:

- F:/git-projects/audio-controller-unity/AudioController/Assets/OSSC/Source/Model/[SoundItem.cs](#)

## 2.4 IPoolable Interface Reference

Used by the [ObjectPool](#)

Inheritance diagram for IPoolable:



## Public Member Functions

- `bool IsFree ()`  
*Checks whether the poolable object is free.*

## Properties

- `PrefabBasedPool pool` [get, set]  
*Saves the pool that it belongs to.*

### 2.4.1 Detailed Description

Used by the [ObjectPool](#)

Definition at line 8 of file [IPoolable.cs](#).

### 2.4.2 Member Function Documentation

#### 2.4.2.1 IsFree()

```
bool IPoolable.IsFree ( )
```

Checks whether the poolable object is free.

#### Returns

True - is Free, False - is busy

Implemented in [OSSC.SoundObject](#).

### 2.4.3 Property Documentation

#### 2.4.3.1 pool

```
PrefabBasedPool IPoolable.pool [get], [set]
```

Saves the pool that it belongs to.

Definition at line 13 of file [IPoolable.cs](#).

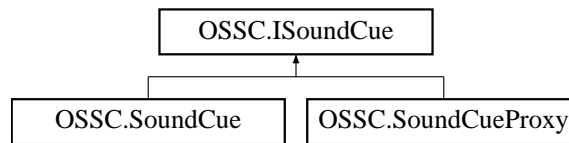
The documentation for this interface was generated from the following file:

- `F:/git-projects/audio-controller-unity/AudioController/Assets/OSSC/Source/IPoolable.cs`

## 2.5 OSSC.ISoundCue Interface Reference

[SoundCue](#) Interface. [SoundController](#) returns a [SoundCue](#) Interface to further control the playing SouncCue

Inheritance diagram for OSSC.ISoundCue:



### Public Member Functions

- void [Play](#) ([SoundCueData](#) data)  
*Plays the [SoundCue](#). This method is called by the [SoundController](#).*
- void [Pause](#) ()  
*Pause the [SoundCue](#).*
- void [Resume](#) ()  
*Resume the paused [SoundCue](#).*
- void [Stop](#) (bool shouldCallOnFinishedCue=true)  
*Stop the [SoundCue](#) from playing.*

### Properties

- Action< string > [OnPlayEnded](#) [get, set]  
*Called everytime a SoundItem finished playing in [SoundCue](#).*
- Action< [SoundCue](#) > [OnPlayCueEnded](#) [get, set]  
*Called everytime a [SoundCue](#) finished playing.*
- [SoundObject](#) [AudioObject](#) [get, set]  
*Used by the [SoundCue](#) to play all SoundItems.*
- [SoundCueData](#) [Data](#) [get]  
*Data collected by the Soundcontroller. Has all SoundItems that needs to be played.*
- bool [IsPlaying](#) [get]  
*Check if [SoundCue](#) is still playing.*
- int [ID](#) [get]  
*[SoundCue](#) Identifier*

### 2.5.1 Detailed Description

[SoundCue](#) Interface. [SoundController](#) returns a [SoundCue](#) Interface to further control the playing SouncCue

Definition at line 12 of file [ISoundCue.cs](#).

### 2.5.2 Member Function Documentation

### 2.5.2.1 Pause()

```
void OSSC.ISoundCue.Pause ( )
```

Pause the [SoundCue](#).

Implemented in [OSSC.SoundCueProxy](#), and [OSSC.SoundCue](#).

### 2.5.2.2 Play()

```
void OSSC.ISoundCue.Play (
    SoundCueData data )
```

Plays the [SoundCue](#). This method is called by the [SoundController](#).

#### Parameters

<i>data</i>	The Data needed for the <a href="#">SoundCue</a> to play.
-------------	---

Implemented in [OSSC.SoundCueProxy](#), and [OSSC.SoundCue](#).

### 2.5.2.3 Resume()

```
void OSSC.ISoundCue.Resume ( )
```

Resume the paused [SoundCue](#).

Implemented in [OSSC.SoundCueProxy](#), and [OSSC.SoundCue](#).

### 2.5.2.4 Stop()

```
void OSSC.ISoundCue.Stop (
    bool shouldCallOnFinishedCue = true )
```

Stop the [SoundCue](#) from playing.

#### Parameters

<i>shouldCallOnFinishedCue</i>	Select whether to Call OnEnd events or not.
--------------------------------	---

Implemented in [OSSC.SoundCueProxy](#), and [OSSC.SoundCue](#).

## 2.5.3 Property Documentation

### 2.5.3.1 AudioObject

[SoundObject](#) OSSC.ISoundCue.AudioObject [get], [set]

Used by the [SoundCue](#) to play all SoundItems.

Definition at line 25 of file [ISoundCue.cs](#).

### 2.5.3.2 Data

[SoundCueData](#) OSSC.ISoundCue.Data [get]

Data collected by the Soundcontroller. Has all SoundItems that needs to be played.

Definition at line 29 of file [ISoundCue.cs](#).

### 2.5.3.3 ID

int OSSC.ISoundCue.ID [get]

[SoundCue](#) Identifier

Definition at line 37 of file [ISoundCue.cs](#).

### 2.5.3.4 IsPlaying

bool OSSC.ISoundCue.IsPlaying [get]

Check if [SoundCue](#) is still playing.

Definition at line 33 of file [ISoundCue.cs](#).

### 2.5.3.5 OnPlayCueEnded

Action<[SoundCue](#)> OSSC.ISoundCue.OnPlayCueEnded [get], [set]

Called everytime a [SoundCue](#) finished playing.

Definition at line 21 of file [ISoundCue.cs](#).

### 2.5.3.6 OnPlayEnded

`Action<string> OSSC.ISoundCue.OnPlayEnded [get], [set]`

Called everytime a SoundItem finished playing in [SoundCue](#).

Definition at line 17 of file [ISoundCue.cs](#).

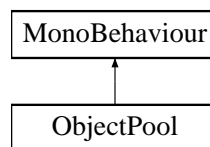
The documentation for this interface was generated from the following file:

- `F:/git-projects/audio-controller-unity/AudioController/Assets/OSSC/Source/ISoundCue.cs`

## 2.6 ObjectPool Class Reference

Creates a pool of different prefabs when someone requests a GameObject.

Inheritance diagram for ObjectPool:



### Public Member Functions

- `GameObject GetFreeObject (GameObject prefab=null)`  
*Gets a Free GameObject.*

### Public Attributes

- `List< PrefabBasedPool > pools`  
*The list of Prefab based pools*

### 2.6.1 Detailed Description

Creates a pool of different prefabs when someone requests a GameObject.

Definition at line 8 of file [ObjectPool.cs](#).

### 2.6.2 Member Function Documentation

#### 2.6.2.1 GetFreeObject()

```

GameObject ObjectPool.GetFreeObject (
    GameObject prefab = null )
  
```

Gets a Free GameObject.

#### Parameters

<i>prefab</i>	The kind of GameObject to return
---------------	----------------------------------

#### Returns

Returns the requested GameObject instance

Definition at line 23 of file [ObjectPool.cs](#).

## 2.6.3 Member Data Documentation

### 2.6.3.1 pools

```
List<PrefabBasedPool> ObjectPool.pools
```

The list of Prefab based pools

Definition at line 14 of file [ObjectPool.cs](#).

The documentation for this class was generated from the following file:

- F:/git-projects/audio-controller-unity/AudioController/Assets/OSSC/Source/[ObjectPool.cs](#)

## 2.7 OSSC.PlaySoundSettings Struct Reference

Set the settings to play a particular cue with particular preferences.

#### Public Member Functions

- void [Init](#) ()

*Initializes the [PlaySoundSettings](#) with predefined values. It is required to be called after the creation of the [PlaySoundSettings](#) instance.*



## Public Attributes

- string [name](#)  
*Name of the soundItem to be played*
- string [] [names](#)  
*A list of sound Items to be played consecutively*
- Transform [parent](#)  
*Attach the Playing sound to a Specific GameObject*
- float [fadeInTime](#)  
*Fade In time of the whole [SoundCue](#)*
- float [fadeOutTime](#)  
*Fade Out time of the whole [SoundCue](#)*
- string [categoryName](#)  
*Play SoundItems from a specific Category*
- bool [isLooped](#)  
*Control whether the [SoundCue](#) should loop*
- [ISoundCue](#) [soundCueProxy](#)  
*Use the same [SoundCue](#) to play again the sounds played in that [SoundCue](#) This is recommended to do, because searching by names all the Sounds to play is very expensive.*
- string [tagName](#)  
*Play soundItems that correspond to the tag*

### 2.7.1 Detailed Description

Set the settings to play a particular cue with particular preferences.

Definition at line 324 of file [SoundController.cs](#).

### 2.7.2 Member Function Documentation

#### 2.7.2.1 Init()

```
void OSSC.PlaySoundSettings.Init ( )
```

Initializes the [PlaySoundSettings](#) with predefined values. It is required to be called after the creation of the [Play↔SoundSettings](#) instance.

Definition at line 368 of file [SoundController.cs](#).

### 2.7.3 Member Data Documentation

#### 2.7.3.1 categoryName

```
string OSSC.PlaySoundSettings.categoryName
```

Play SoundItems from a specific Category

Definition at line 349 of file [SoundController.cs](#).

#### 2.7.3.2 fadeInTime

```
float OSSC.PlaySoundSettings.fadeInTime
```

Fade In time of the whole [SoundCue](#)

Definition at line 341 of file [SoundController.cs](#).

#### 2.7.3.3 fadeOutTime

```
float OSSC.PlaySoundSettings.fadeOutTime
```

Fade Out time of the whole [SoundCue](#)

Definition at line 345 of file [SoundController.cs](#).

#### 2.7.3.4 isLooped

```
bool OSSC.PlaySoundSettings.isLooped
```

Control whether the [SoundCue](#) should loop

Definition at line 353 of file [SoundController.cs](#).

#### 2.7.3.5 name

```
string OSSC.PlaySoundSettings.name
```

Name of the soundItem to be played

Definition at line 329 of file [SoundController.cs](#).

### 2.7.3.6 names

```
string [] OSSC.PlaySoundSettings.names
```

A list of sound Items to be played consecutively

Definition at line 333 of file [SoundController.cs](#).

### 2.7.3.7 parent

```
Transform OSSC.PlaySoundSettings.parent
```

Attach the Playing sound to a Specific GameObject

Definition at line 337 of file [SoundController.cs](#).

### 2.7.3.8 soundCueProxy

```
ISoundCue OSSC.PlaySoundSettings.soundCueProxy
```

Use the same [SoundCue](#) to play again the sounds played in that [SoundCue](#) This is recommended to do, because searching by names all the Sounds to play is very expensive.

Definition at line 358 of file [SoundController.cs](#).

### 2.7.3.9 tagName

```
string OSSC.PlaySoundSettings.tagName
```

Play soundItems that correspond to the tag

Definition at line 362 of file [SoundController.cs](#).

The documentation for this struct was generated from the following file:

- [F:/git-projects/audio-controller-unity/AudioController/Assets/OSSC/Source/SoundController.cs](#)

## 2.8 PrefabBasedPool Class Reference

### Public Member Functions

- [PrefabBasedPool](#) (GameObject [prefab](#))
- GameObject [GetFreeObject](#) ()
- void [Despawn](#) (GameObject obj)

## Public Attributes

- `GameObject` [prefab](#)
- `List< GameObject >` [pool](#)
- `Transform` [parent](#)

*Where pooled objects will reside.*

### 2.8.1 Detailed Description

Definition at line 58 of file [ObjectPool.cs](#).

### 2.8.2 Constructor & Destructor Documentation

#### 2.8.2.1 PrefabBasedPool()

```
PrefabBasedPool.PrefabBasedPool (
    GameObject prefab )
```

Definition at line 60 of file [ObjectPool.cs](#).

### 2.8.3 Member Function Documentation

#### 2.8.3.1 Despawn()

```
void PrefabBasedPool.Despawn (
    GameObject obj )
```

Definition at line 100 of file [ObjectPool.cs](#).

#### 2.8.3.2 GetFreeObject()

```
GameObject PrefabBasedPool.GetFreeObject ( )
```

Definition at line 73 of file [ObjectPool.cs](#).

### 2.8.4 Member Data Documentation

### 2.8.4.1 parent

```
Transform PrefabBasedPool.parent
```

Where pooled objects will reside.

Definition at line 71 of file [ObjectPool.cs](#).

### 2.8.4.2 pool

```
List<GameObject> PrefabBasedPool.pool
```

Definition at line 66 of file [ObjectPool.cs](#).

### 2.8.4.3 prefab

```
GameObject PrefabBasedPool.prefab
```

Definition at line 65 of file [ObjectPool.cs](#).

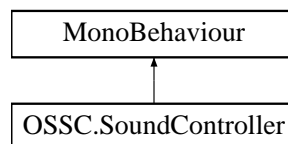
The documentation for this class was generated from the following file:

- F:/git-projects/audio-controller-unity/AudioController/Assets/OSSC/Source/[ObjectPool.cs](#)

## 2.9 OSSC.SoundController Class Reference

The main class that is used for Playing and controlling all sounds.

Inheritance diagram for OSSC.SoundController:



### Public Member Functions

- void [StopAll](#) (bool shouldCallOnEndCallback=true)  
*Stop all Playing Sound Cues.*
- void [SetMute](#) (string categoryName, bool value)  
*Set mute a category.*
- [ISoundCue Play](#) ([PlaySoundSettings](#) settings)  
*Creates a [SoundCue](#) and plays it.*

## Public Attributes

- `GameObject _defaultPrefab`  
Default prefab with [SoundObject](#) and `AudioSource`. It is used by the `Soundcontroller` to play `SoundCues`.
- `SoundControllerData _database`  
Saves all the data that the [SoundController](#) uses.

## Properties

- `GameObject defaultPrefab` [set]  
Set the default Prefab with [SoundObject](#) and `AudioSource` in it.

### 2.9.1 Detailed Description

The main class that is used for Playing and controlling all sounds.

Definition at line 12 of file [SoundController.cs](#).

### 2.9.2 Member Function Documentation

#### 2.9.2.1 Play()

```
ISoundCue OSSC.SoundController.Play (
    PlaySoundSettings settings )
```

Creates a [SoundCue](#) and plays it.

#### Parameters

<i>settings</i>	A struct which contains all data for <a href="#">SoundController</a> to work
-----------------	--

#### Returns

A `soundCue` interface which can be subscribed to it's events.

Definition at line 87 of file [SoundController.cs](#).

#### 2.9.2.2 SetMute()

```
void OSSC.SoundController.SetMute (
    string categoryName,
    bool value )
```

Set mute a category.

## Parameters

<i>categoryName</i>	Name of the cateogory
<i>value</i>	True to mute, false to unmute

Definition at line 71 of file [SoundController.cs](#).

### 2.9.2.3 StopAll()

```
void OSSC.SoundController.StopAll (
    bool shouldCallOnEndCallback = true )
```

Stop all Playing Sound Cues.

## Parameters

<i>shouldCallOnEndCallback</i>	Control whether to call the OnEnd event, or not.
--------------------------------	--

Definition at line 61 of file [SoundController.cs](#).

## 2.9.3 Member Data Documentation

### 2.9.3.1 \_database

[SoundControllerData](#) OSSC.SoundController.\_database

Saves all the data that the [SoundController](#) uses.

Definition at line 23 of file [SoundController.cs](#).

### 2.9.3.2 \_defaultPrefab

GameObject OSSC.SoundController.\_defaultPrefab

Default prefab with [SoundObject](#) and AudioSource. It is used by the Soundcontroller to play SoundCues.

Definition at line 19 of file [SoundController.cs](#).

## 2.9.4 Property Documentation

### 2.9.4.1 defaultPrefab

`GameObject OSSC.SoundController.defaultPrefab [set]`

Set the default Prefab with [SoundObject](#) and AudioSource in it.

Definition at line 50 of file [SoundController.cs](#).

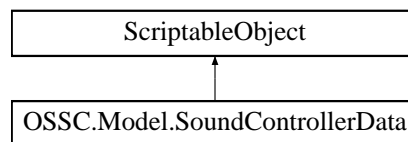
The documentation for this class was generated from the following file:

- `F:/git-projects/audio-controller-unity/AudioController/Assets/OSSC/Source/SoundController.cs`

## 2.10 OSSC.Model.SoundControllerData Class Reference

[SoundController](#)'s Database.

Inheritance diagram for `OSSC.Model.SoundControllerData`:



### Public Attributes

- [CategoryItem](#) [] `items`  
*Stores all created Categories.*
- bool [foldOutCategories](#) = false  
*Checks in editor whether the categories should fold out or not.*
- bool [foldOutTags](#) = false  
*check if editor should fold out the tags or not.*
- string [assetName](#)  
*Database name.*
- [SoundTags](#) `soundTags`  
*Stores the Created tags from [Editor](#).*

### 2.10.1 Detailed Description

[SoundController](#)'s Database.

Definition at line 11 of file [SoundControllerData.cs](#).

### 2.10.2 Member Data Documentation



### 2.10.2.1 assetName

```
string OSSC.Model.SoundControllerData.assetName
```

Database name.

Definition at line 28 of file [SoundControllerData.cs](#).

### 2.10.2.2 foldOutCategories

```
bool OSSC.Model.SoundControllerData.foldOutCategories = false
```

Checks in editor whether the categories should fold out or not.

Definition at line 20 of file [SoundControllerData.cs](#).

### 2.10.2.3 foldOutTags

```
bool OSSC.Model.SoundControllerData.foldOutTags = false
```

check if editor should fold out the tags or not.

Definition at line 24 of file [SoundControllerData.cs](#).

### 2.10.2.4 items

```
CategoryItem [ ] OSSC.Model.SoundControllerData.items
```

Stores all created Categories.

Definition at line 16 of file [SoundControllerData.cs](#).

### 2.10.2.5 soundTags

```
SoundTags OSSC.Model.SoundControllerData.soundTags
```

Stores the Created tags from [Editor](#).

Definition at line 32 of file [SoundControllerData.cs](#).

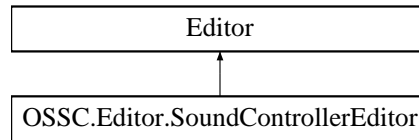
The documentation for this class was generated from the following file:

- F:/git-projects/audio-controller-unity/AudioController/Assets/OSSC/Source/Model/[SoundControllerData.cs](#)

## 2.11 OSSC.Editor.SoundControllerEditor Class Reference

Draws the Custom [Editor](#) for [SoundController](#)

Inheritance diagram for OSSC.Editor.SoundControllerEditor:



### Public Member Functions

- override void [OnInspectorGUI](#) ()  
*Draws the Inspector GUI*

#### 2.11.1 Detailed Description

Draws the Custom [Editor](#) for [SoundController](#)

Definition at line 16 of file [SoundControllerEditor.cs](#).

#### 2.11.2 Member Function Documentation

##### 2.11.2.1 OnInspectorGUI()

```
override void OSSC.Editor.SoundControllerEditor.OnInspectorGUI ( )
```

Draws the Inspector GUI

Definition at line 46 of file [SoundControllerEditor.cs](#).

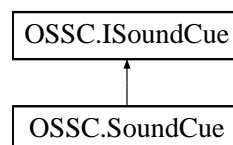
The documentation for this class was generated from the following file:

- F:/git-projects/audio-controller-unity/AudioController/Assets/OSSC/Source/Editor/[SoundControllerEditor.cs](#)

## 2.12 OSSC.SoundCue Class Reference

Plays a whole cue of soundItems

Inheritance diagram for OSSC.SoundCue:



## Public Member Functions

- [SoundCue](#) ()  
*Default Constructor*
- [SoundCue](#) (int id)  
*Custom Constructor*
- void [Play](#) ([SoundCueData](#) data)  
*Will start playing the cue. NOTE: It is called from [SoundCueProxy](#) that is created by the [SoundController](#).*
- void [Play](#) ([SoundCueData](#) data, [SoundCueProxy](#) proxy)  
*Plays the [SoundCue](#).*
- void [Pause](#) ()  
*Will pause the cue;*
- void [Resume](#) ()  
*Resume the cue from where it was paused.*
- void [Stop](#) (bool shouldCallOnFinishedCue=true)  
*Stops the [SoundCue](#).*

## Properties

- Action< string > [OnPlayEnded](#) [get, set]  
*Check [ISoundCue](#)*
- Action< [SoundCue](#) > [OnPlayCueEnded](#) [get, set]  
*Check [ISoundCue](#)*
- Action< [SoundCue](#), [SoundCueProxy](#) > [OnPlayKilled](#) [get, set]  
*Called whenever the sound cue has finished playing or was stopped*
- [SoundObject](#) [AudioObject](#) [get, set]  
*Check [ISoundCue](#)*
- [SoundCueData](#) [Data](#) [get]  
*Check [ISoundCue](#)*
- bool [IsPlaying](#) [get]  
*Check [ISoundCue](#)*
- int [ID](#) [get]  
*[SoundCue](#)'s unique ID given by the manager*

### 2.12.1 Detailed Description

Plays a whole cue of soundItems

Definition at line 12 of file [SoundCue.cs](#).

### 2.12.2 Constructor & Destructor Documentation

### 2.12.2.1 SoundCue() [1/2]

```
OSSC.SoundCue.SoundCue ( )
```

Default Constructor

Definition at line 61 of file [SoundCue.cs](#).

### 2.12.2.2 SoundCue() [2/2]

```
OSSC.SoundCue.SoundCue (
    int id )
```

Custom Constructor

Parameters

<i>id</i>	Sets the ID of the <a href="#">SoundCue</a> .
-----------	---

Definition at line 69 of file [SoundCue.cs](#).

## 2.12.3 Member Function Documentation

### 2.12.3.1 Pause()

```
void OSSC.SoundCue.Pause ( )
```

Will pause the cue;

Implements [OSSC.ISoundCue](#).

Definition at line 118 of file [SoundCue.cs](#).

### 2.12.3.2 Play() [1/2]

```
void OSSC.SoundCue.Play (
    SoundCueData data )
```

Will start playing the cue. NOTE: It is called from [SoundCueProxy](#) that is created by the [SoundController](#).

Implements [OSSC.ISoundCue](#).

Definition at line 91 of file [SoundCue.cs](#).

### 2.12.3.3 Play() [2/2]

```
void OSSC.SoundCue.Play (
    SoundCueData data,
    SoundCueProxy proxy )
```

Plays the [SoundCue](#).

## Parameters

<i>data</i>	<a href="#">SoundCue</a> 's data
<i>proxy</i>	Proxy created by <a href="#">SoundController</a> that called this method.

Definition at line 109 of file [SoundCue.cs](#).

#### 2.12.3.4 Resume()

```
void OSSC.SoundCue.Resume ( )
```

Resume the cue from where it was paused.

Implements [OSSC.ISoundCue](#).

Definition at line 127 of file [SoundCue.cs](#).

#### 2.12.3.5 Stop()

```
void OSSC.SoundCue.Stop (
    bool shouldCallOnFinishedCue = true )
```

Stops the [SoundCue](#).

## Parameters

<i>shouldCallOnFinishedCue</i>	Checks whether to call OnEnd events, or not.
--------------------------------	--

Implements [OSSC.ISoundCue](#).

Definition at line 137 of file [SoundCue.cs](#).

### 2.12.4 Property Documentation

#### 2.12.4.1 AudioObject

```
SoundObject OSSC.SoundCue.AudioObject [get], [set]
```

Check [ISoundCue](#)

Definition at line 32 of file [SoundCue.cs](#).

#### 2.12.4.2 Data

[SoundCueData](#) OSSC.SoundCue.Data [get]

Check [ISoundCue](#)

Definition at line 37 of file [SoundCue.cs](#).

#### 2.12.4.3 ID

int OSSC.SoundCue.ID [get]

[SoundCue](#)'s unique ID given by the manager

##### Returns

Definition at line 53 of file [SoundCue.cs](#).

#### 2.12.4.4 IsPlaying

bool OSSC.SoundCue.IsPlaying [get]

Check [ISoundCue](#)

Definition at line 43 of file [SoundCue.cs](#).

#### 2.12.4.5 OnPlayCueEnded

Action<[SoundCue](#)> OSSC.SoundCue.OnPlayCueEnded [get], [set]

Check [ISoundCue](#)

Definition at line 22 of file [SoundCue.cs](#).

#### 2.12.4.6 OnPlayEnded

Action<string> OSSC.SoundCue.OnPlayEnded [get], [set]

Check [ISoundCue](#)

Definition at line 17 of file [SoundCue.cs](#).

#### 2.12.4.7 OnPlayKilled

Action<[SoundCue](#), [SoundCueProxy](#)> OSSC.SoundCue.OnPlayKilled [get], [set]

Called whenever the sound cue has finished playing or was stopped

Definition at line 27 of file [SoundCue.cs](#).

The documentation for this class was generated from the following file:

- F:/git-projects/audio-controller-unity/AudioController/Assets/OSSC/Source/[SoundCue.cs](#)

## 2.13 OSSC.SoundCueData Struct Reference

Used for sending data to play to AudioCue

### Public Attributes

- [SoundItem](#) [] [sounds](#)  
*sound items that played by the [SoundCue](#).*
- [CategoryItem](#) [] [categoriesForSounds](#)  
*category items that correspond with each of SoundItem in sounds.*
- float [] [categoryVolumes](#)  
*Category sound volumes that correspond with Sound items.*
- GameObject [audioPrefab](#)  
*Prefab with [SoundObject](#) to play Sound items.*
- float [fadeInTime](#)  
*Fade In time.*
- float [fadeOutTime](#)  
*Fade Out time.*
- bool [isFadeIn](#)  
*Should [SoundCue](#) Fade In?*
- bool [isFadeOut](#)  
*Should [SoundCue](#) Fade Out?*
- bool [isLooped](#)  
*Should [SoundCue](#) be looped?*

### 2.13.1 Detailed Description

Used for sending data to play to AudioCue

Definition at line 282 of file [SoundCue.cs](#).

### 2.13.2 Member Data Documentation



### 2.13.2.1 audioPrefab

`GameObject OSSC.SoundCueData.audioPrefab`

Prefab with [SoundObject](#) to play Sound items.

Definition at line 299 of file [SoundCue.cs](#).

### 2.13.2.2 categoriesForSounds

`CategoryItem [ ] OSSC.SoundCueData.categoriesForSounds`

category items that correspond with each of SoundItem in sounds.

Definition at line 291 of file [SoundCue.cs](#).

### 2.13.2.3 categoryVolumes

`float [ ] OSSC.SoundCueData.categoryVolumes`

Category sound volumes that correspond with Sound items.

Definition at line 295 of file [SoundCue.cs](#).

### 2.13.2.4 fadeInTime

`float OSSC.SoundCueData.fadeInTime`

Fade In time.

Definition at line 303 of file [SoundCue.cs](#).

### 2.13.2.5 fadeOutTime

`float OSSC.SoundCueData.fadeOutTime`

Fade Out time.

Definition at line 307 of file [SoundCue.cs](#).

### 2.13.2.6 isFadeIn

```
bool OSSC.SoundCueData.isFadeIn
```

Should [SoundCue](#) Fade In?

Definition at line 311 of file [SoundCue.cs](#).

### 2.13.2.7 isFadeOut

```
bool OSSC.SoundCueData.isFadeOut
```

Should [SoundCue](#) Fade Out?

Definition at line 315 of file [SoundCue.cs](#).

### 2.13.2.8 isLooped

```
bool OSSC.SoundCueData.isLooped
```

Should [SoundCue](#) be looped?

Definition at line 320 of file [SoundCue.cs](#).

### 2.13.2.9 sounds

```
SoundItem [ ] OSSC.SoundCueData.sounds
```

sound items that played by the [SoundCue](#).

Definition at line 287 of file [SoundCue.cs](#).

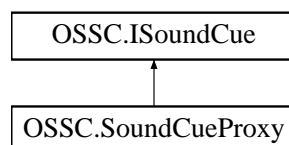
The documentation for this struct was generated from the following file:

- [F:/git-projects/audio-controller-unity/AudioController/Assets/OSSC/Source/SoundCue.cs](#)

## 2.14 OSSC.SoundCueProxy Class Reference

Given by the [SoundController](#) to User as [ISoundCue](#) to control the playing [SoundCue](#).

Inheritance diagram for OSSC.SoundCueProxy:



## Public Member Functions

- void [Play](#) ([SoundCueData](#) data)  
*Check [ISoundCue](#).*
- void [Pause](#) ()  
*Check [ISoundCue](#).*
- void [Resume](#) ()  
*Check [ISoundCue](#).*
- void [Stop](#) (bool shouldCallOnFinishedCue=true)  
*Check [ISoundCue](#).*

## Properties

- [SoundCue](#) [SoundCue](#) [get, set]  
*Sets, Gets the [SoundCue](#).*
- Action< string > [OnPlayEnded](#) [get, set]  
*Check [ISoundCue](#)*
- Action< [SoundCue](#) > [OnPlayCueEnded](#) [get, set]  
*Check [ISoundCue](#)*
- [SoundObject](#) [AudioObject](#) [get, set]  
*Check [ISoundCue](#)*
- [SoundCueData](#) [Data](#) [get]  
*Check [ISoundCue](#)*
- bool [IsPlaying](#) [get]  
*Check [ISoundCue](#)*
- int [ID](#) [get]  
*Check [ISoundCue](#)*

### 2.14.1 Detailed Description

Given by the [SoundController](#) to User as [ISoundCue](#) to control the playing [SoundCue](#).

Definition at line 11 of file [SoundCueProxy.cs](#).

### 2.14.2 Member Function Documentation

#### 2.14.2.1 [Pause\(\)](#)

```
void OSSC.SoundCueProxy.Pause ( )
```

Check [ISoundCue](#).

Implements [OSSC.ISoundCue](#).

Definition at line 127 of file [SoundCueProxy.cs](#).

#### 2.14.2.2 Play()

```
void OSSC.SoundCueProxy.Play (
    SoundCueData data )
```

Check [ISoundCue](#).

Implements [OSSC.ISoundCue](#).

Definition at line 113 of file [SoundCueProxy.cs](#).

#### 2.14.2.3 Resume()

```
void OSSC.SoundCueProxy.Resume ( )
```

Check [ISoundCue](#).

Implements [OSSC.ISoundCue](#).

Definition at line 140 of file [SoundCueProxy.cs](#).

#### 2.14.2.4 Stop()

```
void OSSC.SoundCueProxy.Stop (
    bool shouldCallOnFinishedCue = true )
```

Check [ISoundCue](#).

Implements [OSSC.ISoundCue](#).

Definition at line 153 of file [SoundCueProxy.cs](#).

### 2.14.3 Property Documentation

#### 2.14.3.1 AudioObject

```
SoundObject OSSC.SoundCueProxy.AudioObject [get], [set]
```

Check [ISoundCue](#)

Definition at line 73 of file [SoundCueProxy.cs](#).

### 2.14.3.2 Data

[SoundCueData](#) OSSC.SoundCueProxy.Data [get]

Check [ISoundCue](#)

Definition at line 88 of file [SoundCueProxy.cs](#).

### 2.14.3.3 ID

int OSSC.SoundCueProxy.ID [get]

Check [ISoundCue](#)

Definition at line 104 of file [SoundCueProxy.cs](#).

### 2.14.3.4 IsPlaying

bool OSSC.SoundCueProxy.IsPlaying [get]

Check [ISoundCue](#)

Definition at line 94 of file [SoundCueProxy.cs](#).

### 2.14.3.5 OnPlayCueEnded

Action<[SoundCue](#)> OSSC.SoundCueProxy.OnPlayCueEnded [get], [set]

Check [ISoundCue](#)

Definition at line 58 of file [SoundCueProxy.cs](#).

### 2.14.3.6 OnPlayEnded

Action<string> OSSC.SoundCueProxy.OnPlayEnded [get], [set]

Check [ISoundCue](#)

Definition at line 43 of file [SoundCueProxy.cs](#).

### 2.14.3.7 SoundCue

[SoundCue](#) `OSSC.SoundCueProxy.SoundCue` [get], [set]

Sets, Gets the [SoundCue](#).

Definition at line 28 of file [SoundCueProxy.cs](#).

The documentation for this class was generated from the following file:

- `F:/git-projects/audio-controller-unity/AudioController/Assets/OSSC/Source/SoundCueProxy.cs`

## 2.15 OSSC.Model.SoundItem Class Reference

Used by [CategoryItem](#) to store sounds data.

### Public Attributes

- string [name](#)  
*[SoundItem](#) Name*
- int [tagID](#) = -1  
*Tag ID associated with [SoundItem](#)*
- `UnityEngine.Audio.AudioMixerGroup` [mixer](#)  
*Mixer group associated with this [SoundItem](#).*
- `AudioClip []` [clips](#)  
*List of Audioclips*
- bool [isRandomPitch](#)  
*Is [SoundItem](#) using Random Pitch?*
- `CustomRange` [pitchRange](#) = new `CustomRange()`  
*Range of the Random pitch.*
- bool [isRandomVolume](#)  
*Is [SoundItem](#) using Random Volume?*
- `CustomRange` [volumeRange](#) = new `CustomRange()`  
*Range of the Random Volume.*
- float [volume](#) = 1f  
*Standard volume of the [SoundItem](#)*

### 2.15.1 Detailed Description

Used by [CategoryItem](#) to store sounds data.

Definition at line 9 of file [SoundItem.cs](#).

### 2.15.2 Member Data Documentation

### 2.15.2.1 clips

`AudioClip [] OSSC.Model.SoundItem.clips`

List of Audioclips

Definition at line 26 of file [SoundItem.cs](#).

### 2.15.2.2 isRandomPitch

`bool OSSC.Model.SoundItem.isRandomPitch`

Is [SoundItem](#) using Random Pitch?

Definition at line 30 of file [SoundItem.cs](#).

### 2.15.2.3 isRandomVolume

`bool OSSC.Model.SoundItem.isRandomVolume`

Is [SoundItem](#) using Random Volume?

Definition at line 38 of file [SoundItem.cs](#).

### 2.15.2.4 mixer

`UnityEngine.Audio.AudioMixerGroup OSSC.Model.SoundItem.mixer`

Mixer group associated with this [SoundItem](#).

Definition at line 22 of file [SoundItem.cs](#).

### 2.15.2.5 name

`string OSSC.Model.SoundItem.name`

[SoundItem](#) Name

Definition at line 14 of file [SoundItem.cs](#).

#### 2.15.2.6 pitchRange

```
CustomRange OSSC.Model.SoundItem.pitchRange = new CustomRange()
```

Range of the Random pitch.

Definition at line 34 of file [SoundItem.cs](#).

#### 2.15.2.7 tagID

```
int OSSC.Model.SoundItem.tagID = -1
```

Tag ID associated with [SoundItem](#)

Definition at line 18 of file [SoundItem.cs](#).

#### 2.15.2.8 volume

```
float OSSC.Model.SoundItem.volume = 1f
```

Standard volume of the [SoundItem](#)

Definition at line 48 of file [SoundItem.cs](#).

#### 2.15.2.9 volumeRange

```
CustomRange OSSC.Model.SoundItem.volumeRange = new CustomRange()
```

Range of the Random Volume.

Definition at line 42 of file [SoundItem.cs](#).

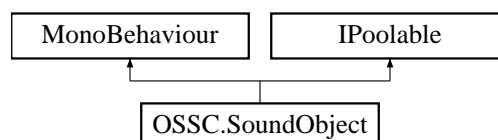
The documentation for this class was generated from the following file:

- F:/git-projects/audio-controller-unity/AudioController/Assets/OSSC/Source/Model/[SoundItem.cs](#)

## 2.16 OSSC.SoundObject Class Reference

Used by the [SoundCue](#). Controls the AudioSource.

Inheritance diagram for OSSC.SoundObject:





## Public Member Functions

- void [Setup](#) (string id, AudioClip clip, float volume, float fadeInTime=0f, float fadeOutTime=0f, AudioManager↵ Group mixer=null, float pitch=1f)  
*Prepares the [SoundObject](#) for playing an AudioClip.*
- void [Play](#) ()  
*Plays the AudioSource.*
- void [Pause](#) ()  
*Pauses the AudioSource.*
- void [Resume](#) ()  
*Resumes from Pause.*
- void [Stop](#) ()  
*Stops the [SoundObject](#) from playing.*
- bool [IsFree](#) ()  
*Check [IPoolable](#)*

## Public Attributes

- System.Action< [SoundObject](#) > [OnFinishedPlaying](#)  
*Called when [SoundObject](#) finishes playing.*

## Properties

- bool [isDespawnOnFinishedPlaying](#) [get, set]  
*Check whether [SoundObject](#) should despawn after finishing playing.*
- string [clipName](#) [get]  
*AudioClip name played.*
- AudioSource [source](#) [get]  
*Gets the [SoundObject](#)'s AudioSource.*
- string [ID](#) [get]  
*Gets the [SoundObject](#)'s ID.*

### 2.16.1 Detailed Description

Used by the [SoundCue](#). Controls the AudioSource.

Definition at line 14 of file [SoundObject.cs](#).

### 2.16.2 Member Function Documentation

#### 2.16.2.1 IsFree()

```
bool OSSC.SoundObject.IsFree ( )
```

Check [IPoolable](#)

Implements [IPoolable](#).

Definition at line 291 of file [SoundObject.cs](#).

#### 2.16.2.2 Pause()

```
void OSSC.SoundObject.Pause ( )
```

Pauses the AudioSource.

Definition at line 153 of file [SoundObject.cs](#).

#### 2.16.2.3 Play()

```
void OSSC.SoundObject.Play ( )
```

Plays the AudioSource.

Definition at line 137 of file [SoundObject.cs](#).

#### 2.16.2.4 Resume()

```
void OSSC.SoundObject.Resume ( )
```

Resumes from Pause.

Definition at line 164 of file [SoundObject.cs](#).

#### 2.16.2.5 Setup()

```
void OSSC.SoundObject.Setup (
    string id,
    AudioClip clip,
    float volume,
    float fadeInTime = 0f,
    float fadeOutTime = 0f,
    AudioManager mixer = null,
    float pitch = 1f )
```

Prepares the [SoundObject](#) for playing an AudioClip.

## Parameters

<i>id</i>	<a href="#">SoundObject</a> 's ID
<i>clip</i>	AudioClip to play
<i>volume</i>	volume of the sound.
<i>fadeInTime</i>	Fade In Time
<i>fadeOutTime</i>	Fade Out Time
<i>mixer</i>	Audio Mixer group
<i>pitch</i>	Pitch of the sound

Definition at line 117 of file [SoundObject.cs](#).

### 2.16.2.6 Stop()

```
void OSSC.SoundObject.Stop ( )
```

Stops the [SoundObject](#) from playing.

Definition at line 175 of file [SoundObject.cs](#).

## 2.16.3 Member Data Documentation

### 2.16.3.1 OnFinishedPlaying

```
System.Action<SoundObject> OSSC.SoundObject.OnFinishedPlaying
```

Called when [SoundObject](#) finishes playing.

Definition at line 20 of file [SoundObject.cs](#).

## 2.16.4 Property Documentation

### 2.16.4.1 clipName

```
string OSSC.SoundObject.clipName [get]
```

AudioClip name played.

Definition at line 85 of file [SoundObject.cs](#).

#### 2.16.4.2 ID

```
string OSSC.SoundObject.ID [get]
```

Gets the [SoundObject](#)'s ID.

Definition at line 103 of file [SoundObject.cs](#).

#### 2.16.4.3 isDespawnOnFinishedPlaying

```
bool OSSC.SoundObject.isDespawnOnFinishedPlaying [get], [set]
```

Check whether [SoundObject](#) should despawn after finishing playing.

Definition at line 76 of file [SoundObject.cs](#).

#### 2.16.4.4 source

```
AudioSource OSSC.SoundObject.source [get]
```

Gets the [SoundObject](#)'s AudioSource.

Definition at line 96 of file [SoundObject.cs](#).

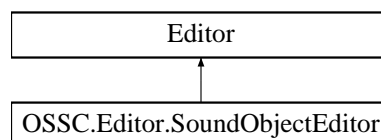
The documentation for this class was generated from the following file:

- F:/git-projects/audio-controller-unity/AudioController/Assets/OSSC/Source/[SoundObject.cs](#)

## 2.17 OSSC.Editor.SoundObjectEditor Class Reference

Draw the custom editor inspector for [SoundObject](#)

Inheritance diagram for OSSC.Editor.SoundObjectEditor:



### Public Member Functions

- override void [OnInspectorGUI](#) ()  
*Draws the inspector's GUI*

### 2.17.1 Detailed Description

Draw the custom editor inspector for [SoundObject](#)

Definition at line 13 of file [SoundObjectEditor.cs](#).

### 2.17.2 Member Function Documentation

#### 2.17.2.1 OnInspectorGUI()

```
override void OSSC.Editor.SoundObjectEditor.OnInspectorGUI ( )
```

Draws the inspector's GUI

Definition at line 27 of file [SoundObjectEditor.cs](#).

The documentation for this class was generated from the following file:

- [F:/git-projects/audio-controller-unity/AudioController/Assets/OSSC/Source/Editor/SoundObjectEditor.cs](#)

## 2.18 OSSC.SoundTags Class Reference

Used By the [SoundController](#) for tagging SoundItems.

### Public Member Functions

- [SoundTags](#) ()  
*Default Constructor*
- [TagData \[\] ToArray](#) ()  
*Returns all data in form of an array*
- [string \[\] ToArrayNames](#) ()  
*Returns the names of the tags.*
- [int \[\] ToArrayIDs](#) ()  
*Returns the IDs of the tags.*
- [TagData GetTagDataByName](#) (string name)  
*Gets TagData by name.*
- [TagData GetTagDataByID](#) (int ID)  
*Gets TagData by ID.*
- [int GetTagIDByName](#) (string name)  
*Gets Tag ID by name.*
- [string GetTagNameByID](#) (int ID)  
*Gets Tag name by ID*
- [void SetTag](#) (string name)  
*Sets a new Tag.*
- [void RemoveByTag](#) ([TagData](#) data)  
*Removes a Tag by TagData.*

### 2.18.1 Detailed Description

Used By the [SoundController](#) for tagging SoundItems.

Definition at line 10 of file [SoundTags.cs](#).

### 2.18.2 Constructor & Destructor Documentation

#### 2.18.2.1 SoundTags()

```
OSSC.SoundTags.SoundTags ( )
```

Default Constructor

Definition at line 25 of file [SoundTags.cs](#).

### 2.18.3 Member Function Documentation

#### 2.18.3.1 GetTagDataByID()

```
TagData OSSC.SoundTags.GetTagDataByID (
    int ID )
```

Gets [TagData](#) by ID.

##### Parameters

<i>ID</i>	ID of the Tag.
-----------	----------------

##### Returns

[TagData](#) with the corresponding ID.

Definition at line 86 of file [SoundTags.cs](#).

#### 2.18.3.2 GetTagDataByName()

```
TagData OSSC.SoundTags.GetTagDataByName (
    string name )
```

Gets [TagData](#) by name.

**Parameters**

<i>name</i>	name of the Tag.
-------------	------------------

**Returns**

[TagData](#) with the corresponding name.

Definition at line 76 of file [SoundTags.cs](#).

**2.18.3.3 GetTagIDByName()**

```
int OSSC.SoundTags.GetTagIDByName (
    string name )
```

Gets Tag ID by name.

**Parameters**

<i>name</i>	name of the Tag.
-------------	------------------

**Returns**

ID of the Tag.

Definition at line 96 of file [SoundTags.cs](#).

**2.18.3.4 GetTagNameByID()**

```
string OSSC.SoundTags.GetTagNameByID (
    int ID )
```

Gets Tag name by ID

**Parameters**

<i>ID</i>	ID of the Tag
-----------	---------------

**Returns**

Name of the Tag.

Definition at line 110 of file [SoundTags.cs](#).

#### 2.18.3.5 RemoveByTag()

```
void OSSC.SoundTags.RemoveByTag (
    TagData data )
```

Removes a Tag by [TagData](#).

##### Parameters

<i>data</i>	<a href="#">TagData</a> that wants to be removed.
-------------	---

Definition at line [143](#) of file [SoundTags.cs](#).

#### 2.18.3.6 SetTag()

```
void OSSC.SoundTags.SetTag (
    string name )
```

Sets a new Tag.

##### Parameters

<i>name</i>	Name of the Tag.
-------------	------------------

Definition at line [123](#) of file [SoundTags.cs](#).

#### 2.18.3.7 ToArray()

```
TagData [ ] OSSC.SoundTags.ToArray ( )
```

Returns all data in form of an array

##### Returns

Array of [TagData](#).

Definition at line [36](#) of file [SoundTags.cs](#).

#### 2.18.3.8 ToArrayIDs()

```
int [ ] OSSC.SoundTags.ToArrayIDs ( )
```

Returns the IDs of the tags.

##### Returns

int[] with ids

Definition at line [60](#) of file [SoundTags.cs](#).



### 2.18.3.9 ToArrayNames()

```
string [] OSSC.SoundTags.ToArrayNames ( )
```

Returns the names of the tags.

#### Returns

string[] with names

Definition at line 45 of file [SoundTags.cs](#).

The documentation for this class was generated from the following file:

- F:/git-projects/audio-controller-unity/AudioController/Assets/OSSC/Source/[SoundTags.cs](#)

## 2.19 OSSC.TagData Struct Reference

Used by the [SoundTags](#) to save Tags.

### Public Attributes

- string [name](#)  
*Tag Name*
- int [ID](#)  
*Tag ID*

### 2.19.1 Detailed Description

Used by the [SoundTags](#) to save Tags.

Definition at line 154 of file [SoundTags.cs](#).

### 2.19.2 Member Data Documentation

#### 2.19.2.1 ID

```
int OSSC.TagData.ID
```

Tag ID

Definition at line 163 of file [SoundTags.cs](#).

#### 2.19.2.2 name

```
string OSSC.TagData.name
```

Tag Name

Definition at line 159 of file [SoundTags.cs](#).

The documentation for this struct was generated from the following file:

- F:/git-projects/audio-controller-unity/AudioController/Assets/OSSC/Source/[SoundTags.cs](#)



## Chapter 3

# File Documentation

### 3.1 F:/git-projects/audio-controller-unity/AudioController/Assets/OSSC/Source/CueManager.cs File Reference

#### Classes

- class [OSSC.CueManager](#)  
*Manages all SoundCues*

#### Namespaces

- namespace [OSSC](#)

### 3.2 CueManager.cs

```
00001 using System.Collections;
00002 using System.Collections.Generic;
00003 using UnityEngine;
00004
00005 namespace OSSC
00006 {
00010     public class CueManager
00011     {
00012         #region Private fields
00013         private List<SoundCue> _soundCues;
00017         #endregion
00018
00019         #region Public Methods and Properties
00020         public CueManager()
00024         {
00025             _soundCues = new List<SoundCue>();
00026         }
00027
00032         public CueManager(int initialSize)
00033         {
00034             _soundCues = new List<SoundCue>(initialSize);
00035         }
00036
00041         public SoundCue GetSoundCue()
00042         {
00043             SoundCue cue = FindFreeCue();
00044             cue.OnPlayKilled += OnPlayKilled_handler;
00045             return cue;
00046         }
00047
00052         public void StopAllCues(bool shouldCallOnEndCallback = true)
```

```

00053     {
00054         for (int i = 0; i < _soundCues.Count; i++)
00055         {
00056             if (_soundCues[i].IsPlaying)
00057                 _soundCues[i].Stop(shouldCallOnEndCallback);
00058         }
00059     }
00060     #endregion
00061
00062     #region Private methods
00063     private void OnPlayKilled_handler(SoundCue cue, SoundCueProxy proxy)
00064     {
00065         //NOTE: Clear up any references to events.
00066         cue.OnPlayKilled = null;
00067         cue.OnPlayCueEnded = null;
00068         cue.OnPlayEnded = null;
00069         proxy.SoundCue = null;
00070     }
00071
00072     private SoundCue FindFreeCue()
00073     {
00074         SoundCue cue = null;
00075         for (int i = 0; i < _soundCues.Count; i++)
00076         {
00077             if (_soundCues[i].IsPlaying == false)
00078             {
00079                 cue = _soundCues[i];
00080                 break;
00081             }
00082         }
00083
00084         if (cue == null)
00085         {
00086             cue = new SoundCue(_soundCues.Count);
00087             _soundCues.Add(cue);
00088         }
00089
00090         return cue;
00091     }
00092     #endregion
00093 }
00094
00095
00096
00097
00098

```

### 3.3 F:/git-projects/audio-controller-unity/AudioController/Assets/OSSC/Source/Editor/SoundControllerEditor.cs File Reference

#### Classes

- class [OSSC.Editor.SoundControllerEditor](#)  
*Draws the Custom Editor for SoundController*

#### Namespaces

- namespace [OSSC.Editor](#)

### 3.4 SoundControllerEditor.cs

```

00001 using System.Collections;
00002 using System.Collections.Generic;
00003 using System.Security.Cryptography;
00004 using UnityEngine;
00005 using UnityEditor;
00006 using OSSC.Model;
00007 using UnityEngine.Audio;
00008 using UnityEngine.EventSystems;
00009
00010 namespace OSSC.Editor
00011 {

```

```

00015     [CustomEditor(typeof(SoundController))]
00016     public class SoundControllerEditor : UnityEditor.Editor
00017     {
00021         private const int NAME_ABV_LEN = 50;
00025         private const float PITCH_RANGE_MAX = 3f;
00029         private const float PITCH_RANGE_MIN = -3f;
00033         private SoundController _ac;
00037         private string categoryNameSearch = "";
00041         private string _tagName = "";
00042
00046         public override void OnInspectorGUI()
00047         {
00048             base.OnInspectorGUI();
00049             _ac = target as SoundController;
00050
00051             if (_ac._database == null)
00052             {
00053                 EditorGUILayout.HelpBox("Create SoundControllerData asset, then throw it here.",
MessageTypes.Info);
00054             }
00055             else
00056             {
00057                 DrawMain();
00058             }
00059
00060             EditorUtility.SetDirty(_ac);
00061             if (_ac._database != null)
00062                 EditorUtility.SetDirty(_ac._database);
00063         }
00064
00068         private void DrawMain()
00069         {
00070             if (_ac._database == null)
00071                 return;
00072             DrawSoundTags();
00073             EditorGUILayout.BeginHorizontal(EditorStyles.helpBox);
00074             if (GUILayout.Button("DELETE DATA"))
00075             {
00076                 AssetDatabase.DeleteAsset(AssetDatabase.GetAssetPath(_ac.
_database));
00077                 return;
00078             }
00079             var db = _ac._database;
00080             if (GUILayout.Button("ADD CATEGORY"))
00081             {
00082                 var category = new Model.CategoryItem();
00083                 var categories = new Model.CategoryItem[db.items != null ? db.items.Length + 1 : 1];
00084                 if (db.items != null)
00085                     db.items.CopyTo(categories, 0);
00086                 categories[categories.Length - 1] = category;
00087                 db.items = categories;
00088             }
00089             EditorGUILayout.EndHorizontal();
00090             DrawCategories(_ac._database);
00091         }
00092
00097         private void DrawCategories(Model.SoundControllerData db)
00098         {
00099
00100             if (db.items == null)
00101                 return;
00102             if (db.items.Length == 0)
00103                 return;
00104
00105             categoryNameSearch = EditorGUILayout.TextField("Search Category", categoryNameSearch);
00106             db.foldOutCategories = EditorGUILayout.Foldout(db.foldOutCategories, "CATEGORIES", true);
00107             if (!db.foldOutCategories)
00108                 return;
00109
00110             for (int i = db.items.Length - 1; i >= 0; i--)
00111             {
00112                 if (!string.IsNullOrEmpty(db.items[i].name))
00113                     if (db.items[i].name.ToLower().Contains(categoryNameSearch.ToLower()) == false &&
string.IsNullOrEmpty(categoryNameSearch) == false)
00114                         continue;
00115                 DrawCategory(db.items[i], i);
00116             }
00117         }
00118
00124         private void DrawCategory(Model.CategoryItem item, int index)
00125         {
00126             EditorGUILayout.BeginVertical(EditorStyles.helpBox);
00127
00128             item.name = EditorGUILayout.TextField("Name", item.name);
00129             item.audioObjectPrefab = (GameObject)EditorGUILayout.ObjectField("Category AO prefab", item.
audioObjectPrefab, typeof(GameObject), false);
00130             item.usingDefaultPrefab = item.audioObjectPrefab == null;

```

```

00131         item.isMute = EditorGUILayout.Toggle("Is Mute", item.isMute);
00132         item.categoryVolume = EditorGUILayout.Slider("Category Volume", item.categoryVolume, 0f, 1f);
00133
00134         EditorGUILayout.BeginHorizontal(EditorStyles.helpBox);
00135         if (GUILayout.Button("ADD SOUND ITEM"))
00136         {
00137             var soundItem = new Model.SoundItem();
00138             bool isNoSoundItems = item.soundItems == null;
00139             var soundItems = new Model.SoundItem[isNoSoundItems ? item.soundItems.Length + 1 : 1];
00140             if (!isNoSoundItems)
00141                 item.soundItems.CopyTo(soundItems, 0);
00142             soundItems[soundItems.Length - 1] = soundItem;
00143             item.soundItems = soundItems;
00144         }
00145         string nameAbv = "";
00146         if (string.IsNullOrEmpty(item.name) == false)
00147             nameAbv = item.name.Length > NAME_ABV_LEN ? item.name.Substring(0, NAME_ABV_LEN) : item.
name;
00148         if (GUILayout.Button("Delete " + nameAbv))
00149         {
00150             DeleteCategory(index);
00151         }
00152         EditorGUILayout.EndHorizontal();
00153
00154         if (item.soundItems != null)
00155             if (item.soundItems.Length != 0)
00156                 item.soundsSearchName = EditorGUILayout.TextField("Search sound item", item.
soundsSearchName);
00157
00158         item.foldOutSoundItems = DrawSoundItems(item, item.foldOutSoundItems, item.soundsSearchName);
00159
00160         EditorGUILayout.EndVertical();
00161     }
00162
00163     private void DeleteCategory(int index)
00164     {
00165         var categories = new Model.CategoryItem[_ac._database.items.Length - 1];
00166         int catInd = 0;
00167         for (int i = 0; i < _ac._database.items.Length; i++)
00168         {
00169             if (i == index)
00170                 continue;
00171
00172             categories[catInd] = _ac._database.items[i];
00173             catInd += 1;
00174         }
00175         _ac._database.items = categories;
00176     }
00177
00178     private bool DrawSoundItems(Model.CategoryItem item, bool foldOut, string searchName)
00179     {
00180         Model.SoundItem[] items = item.soundItems;
00181         if (items == null || items.Length == 0)
00182             return foldOut;
00183
00184         EditorGUI.indentLevel++;
00185
00186         EditorGUILayout.BeginHorizontal();
00187         foldOut = EditorGUILayout.Foldout(foldOut, "SOUND ITEMS", true);
00188         if (items != null)
00189         {
00190             if (items.Length != 0)
00191                 if (GUILayout.Button("DELETE ALL SOUNDS"))
00192                 {
00193                     items = new Model.SoundItem[0];
00194                     item.soundItems = items;
00195                     return foldOut;
00196                 }
00197         }
00198         EditorGUILayout.EndHorizontal();
00199
00200         if (foldOut)
00201         {
00202             for (int j = items.Length - 1; j >= 0; j--)
00203             {
00204                 if (!string.IsNullOrEmpty(items[j].name))
00205                     if (items[j].name.ToLower().Contains(searchName.ToLower()) == false && string.
IsNullOrEmpty(searchName) == false)
00206                         continue;
00207                 DrawSoundItem(items[j], j, items);
00208             }
00209         }
00210         EditorGUI.indentLevel--;
00211         return foldOut;
00212     }
00213
00214     private void DrawSoundItem(Model.SoundItem item, int index, Model.SoundItem[] items)

```

```

00232     {
00233         EditorGUILayout.BeginVertical(EditorStyles.helpBox);
00234         item.name = EditorGUILayout.TextField("Name", item.name);
00235
00236         string[] names = _ac._database.soundTags.
ToArrayNames();
00237         int[] ids = _ac._database.soundTags.ToArrayIDs();
00238         if (ids.Length != 0)
00239         {
00240             int indexTag = System.Array.IndexOf(ids, item.tagID);
00241             indexTag = EditorGUILayout.Popup("Tag", indexTag, names);
00242             if (indexTag != -1)
00243                 item.tagID = ids[indexTag];
00244         }
00245         else
00246         {
00247             item.tagID = -1;
00248         }
00249
00250
00251         item.mixer = (AudioMixerGroup)EditorGUILayout.ObjectField("Mixer", item.mixer, typeof(
AudioMixerGroup), false);
00252         if (item.clips == null)
00253         {
00254             item.clips = new AudioClip[1];
00255         }
00256         int size = item.clips.Length;
00257         size = EditorGUILayout.IntField("Size", size);
00258         if (size != item.clips.Length)
00259         {
00260             var newClips = new AudioClip[size];
00261             for (int i = 0; i < item.clips.Length; i++)
00262             {
00263                 if (i >= size)
00264                     break;
00265                 newClips[i] = item.clips[i];
00266             }
00267             item.clips = newClips;
00268         }
00269         for (int i = 0; i < item.clips.Length; i++)
00270         {
00271             item.clips[i] = (AudioClip)EditorGUILayout.ObjectField(item.clips[i], typeof(AudioClip),
false);
00272         }
00273
00274         item.isRandomVolume =
00275             EditorGUILayout.ToggleLeft("Use Random Volume", item.isRandomVolume, EditorStyles.boldLabel
);
00276         if (!item.isRandomVolume)
00277             item.volume = EditorGUILayout.Slider("Volume", item.volume, 0f, 1f);
00278         else
00279         {
00280             EditorGUILayout.LabelField("Min Volume:", item.volumeRange.min.ToString(), EditorStyles.
largeLabel);
00281             EditorGUILayout.LabelField("Max Volume:", item.volumeRange.max.ToString(), EditorStyles.
largeLabel);
00282             EditorGUILayout.MinMaxSlider("Volume Range", ref item.volumeRange.min, ref item.volumeRange
.max, 0f, 1f);
00283         }
00284
00285         item.isRandomPitch =
00286             EditorGUILayout.ToggleLeft("Use Random Pitch", item.isRandomPitch, EditorStyles.boldLabel);
00287         if (item.isRandomPitch)
00288         {
00289             EditorGUILayout.LabelField("Min Pitch:", item.pitchRange.min.ToString(), EditorStyles.
largeLabel);
00290             EditorGUILayout.LabelField("Max Pitch:", item.pitchRange.max.ToString(), EditorStyles.
largeLabel);
00291             EditorGUILayout.MinMaxSlider("Pitch Range", ref item.pitchRange.min, ref item.pitchRange.
max, PITCH_RANGE_MIN, PITCH_RANGE_MAX);
00292         }
00293         string nameAbv = "";
00294         if (string.IsNullOrEmpty(item.name) == false)
00295             nameAbv = item.name.Length > NAME_ABV_LEN ? item.name.Substring(0, NAME_ABV_LEN) : item.
name;
00296         if (GUILayout.Button("Delete Item " + nameAbv))
00297         {
00298             DeleteSoundItem(index, items);
00299         }
00300         EditorGUILayout.EndVertical();
00301     }
00302
00303     private void DeleteSoundItem(int index, Model.SoundItem[] items)
00304     {
00305         var category = System.Array.Find(_ac._database.items, (x) => {
00306             return x.soundItems == items;
00307         });
00308     }

```

```

00313         var soundItems = new Model.SoundItem[category.soundItems.Length - 1];
00314         int soundInd = 0;
00315         for (int i = 0; i < category.soundItems.Length; i++)
00316         {
00317             if (i == index)
00318                 continue;
00319             soundItems[soundInd] = category.soundItems[i];
00320             soundInd += 1;
00321         }
00322         category.soundItems = soundItems;
00323     }
00324
00325     private void DrawSoundTags()
00326     {
00327         if (_ac._database.soundTags == null)
00328         {
00329             _ac._database.soundTags = new SoundTags();
00330         }
00331         _ac._database.foldOutTags = EditorGUILayout.Foldout(_ac.
00332 _database.foldOutTags, "Tags", true);
00333         if (_ac._database.foldOutTags == false)
00334         {
00335             EditorGUILayout.HelpBox("Add tags filter sounds by them.", MessageType.Info);
00336             return;
00337         }
00338         EditorGUILayout.BeginVertical(EditorStyles.helpBox);
00339         DrawAddNewTag();
00340
00341         TagData[] data = _ac._database.soundTags.
00342     ToArray();
00343         for (int i = 0; i < data.Length; i++)
00344         {
00345             DrawSoundTag(data[i], _ac._database.soundTags);
00346         }
00347         EditorGUILayout.EndVertical();
00348     }
00349
00350     private void DrawSoundTag(TagData data, SoundTags tags)
00351     {
00352         EditorGUILayout.BeginHorizontal();
00353         EditorGUILayout.LabelField("ID: " + data.ID.ToString(), "name: " + data.
00354 name);
00355         if (GUILayout.Button("Delete"))
00356         {
00357             tags.RemoveByTag(data);
00358         }
00359         EditorGUILayout.EndHorizontal();
00360     }
00361
00362     private void DrawAddNewTag()
00363     {
00364         EditorGUILayout.BeginHorizontal(EditorStyles.helpBox);
00365         _tagName = EditorGUILayout.TextField("Add Tag:", _tagName);
00366         if (GUILayout.Button("Add"))
00367         {
00368             _ac._database.soundTags.SetTag(_tagName);
00369             _tagName = string.Empty;
00370         }
00371         EditorGUILayout.EndHorizontal();
00372     }
00373 }
00374
00375 }
00376
00377 }
00378
00379 }
00380
00381 }
00382
00383 }
00384
00385 }

```

### 3.5 F:/git-projects/audio-controller-unity/AudioController/Assets/OSSC/Source/Editor/SoundObjectEditor.cs File Reference

#### Classes

- class [OSSC.Editor.SoundObjectEditor](#)  
Draw the custom editor inspector for [SoundObject](#)

#### Namespaces

- namespace [OSSC.Editor](#)



## 3.6 SoundObjectEditor.cs

```

00001 using System.Collections;
00002 using System.Collections.Generic;
00003 using UnityEngine;
00004 using UnityEditor;
00005 using OSSC;
00006
00007 namespace OSSC.Editor
00008 {
00012     [CustomEditor(typeof(SoundObject))]
00013     public class SoundObjectEditor : UnityEditor.Editor
00014     {
00018         private SoundObject _ao;
00022         private bool _showControls = false;
00023
00027         public override void OnInspectorGUI()
00028         {
00029             _ao = target as SoundObject;
00030
00031             EditorGUILayout.LabelField("Sound Item", _ao.ID, EditorStyles.boldLabel);
00032             EditorGUILayout.LabelField("Current Clip", _ao.clipName);
00033             _showControls = EditorGUILayout.ToggleLeft("Show Controls", _showControls);
00034             if (_showControls == false)
00035                 return;
00036
00037             ShowControls();
00038         }
00039
00043         private void ShowControls()
00044         {
00045             if (_ao.source.isPlaying)
00046             {
00047                 if (GUILayout.Button("Stop"))
00048                 {
00049                     _ao.Stop();
00050                 }
00051             }
00052
00053             if (_ao.source.clip != null)
00054             {
00055                 int minutes = (int)(_ao.source.time / 60f);
00056                 int seconds = (int)(_ao.source.time - minutes * 60);
00057                 EditorGUILayout.LabelField("Current Time", minutes + ":" + seconds);
00058                 minutes = (int)(_ao.source.clip.length / 60f);
00059                 seconds = (int)(_ao.source.clip.length - minutes * 60);
00060                 EditorGUILayout.LabelField("Clip Time", minutes + ":" + seconds);
00061                 _ao.source.time = EditorGUILayout.Slider("Seek", _ao.
source.time, 0f, _ao.source.clip.length);
00062                 Repaint();
00063             }
00064         }
00065     }
00066 }

```

## 3.7 F:/git-projects/audio-controller-unity/AudioController/Assets/OSSC/Source/IPoolable.cs File Reference

### Classes

- interface [IPoolable](#)  
*Used by the [ObjectPool](#)*

## 3.8 IPoolable.cs

```

00001 using System.Collections;
00002 using System.Collections.Generic;
00003 using UnityEngine;
00004
00008 public interface IPoolable
00009 {

```

```

00013     PrefabBasedPool pool {
00014         get; set;
00015     }
00016
00021     bool IsFree();
00022 }

```

### 3.9 F:/git-projects/audio-controller-unity/AudioController/Assets/OSSC/Source/ISoundCue.cs File Reference ↩

#### Classes

- interface [OSSC.ISoundCue](#)  
*SoundCue Interface. SoundController returns a SoundCue Interface to further control the playing SoundCue*

#### Namespaces

- namespace [OSSC](#)

### 3.10 ISoundCue.cs

```

00001 using System;
00002 using System.Collections;
00003 using System.Collections.Generic;
00004 using UnityEngine;
00005
00006 namespace OSSC
00007 {
00012     public interface ISoundCue
00013     {
00017         Action<string> OnPlayEnded { get; set; }
00021         Action<SoundCue> OnPlayCueEnded { get; set; }
00025         SoundObject AudioObject { get; set; }
00029         SoundCueData Data { get; }
00033         bool IsPlaying { get; }
00037         int ID { get; }
00038
00044         void Play(SoundCueData data);
00048         void Pause();
00052         void Resume();
00057         void Stop(bool shouldCallOnFinishedCue = true);
00058     }
00059 }

```

### 3.11 F:/git-projects/audio-controller-unity/AudioController/Assets/OSSC/Source/Model/CategoryItem.cs File Reference ↩

#### Classes

- class [OSSC.Model.CategoryItem](#)  
*Used by the SoundControllerData to store categories.*

#### Namespaces

- namespace [OSSC.Model](#)

## 3.12 CategoryItem.cs

```

00001 using System.Collections;
00002 using System.Collections.Generic;
00003 using UnityEngine;
00004
00005 namespace OSSC.Model
00006 {
00010     [System.Serializable]
00011     public class CategoryItem
00012     {
00016         public string name;
00020         public SoundItem[] soundItems;
00024         public GameObject audioObjectPrefab;
00028         public bool usingDefaultPrefab = true;
00029
00033         [Range(0f, 1f)]
00034         public float categoryVolume = 1f;
00035
00039         public bool foldOutSoundItems = false;
00043         public string soundsSearchName = "";
00047         public bool isMute = false;
00048     }
00049
00050 }

```

## 3.13 F:/git-projects/audio-controller-unity/AudioController/Assets/OSSC/Source/Model/SoundControllerData.cs File Reference ↩

### Classes

- class [OSSC.Model.SoundControllerData](#)  
*SoundController's Database.*

### Namespaces

- namespace [OSSC.Model](#)

## 3.14 SoundControllerData.cs

```

00001 using System.Collections;
00002 using System.Collections.Generic;
00003 using UnityEngine;
00004
00005 namespace OSSC.Model
00006 {
00010     [CreateAssetMenu(fileName = "NewSoundControllerData", menuName = "Sound Controller/New
    SoundControllerData")]
00011     public class SoundControllerData : ScriptableObject
00012     {
00016         public CategoryItem[] items;
00020         public bool foldOutCategories = false;
00024         public bool foldOutTags = false;
00028         public string assetName;
00032         public SoundTags soundTags;
00033     }
00034 }

```

### 3.15 F:/git-projects/audio-controller-unity/AudioController/Assets/OSSC/Source/Model/↔ SoundItem.cs File Reference

#### Classes

- class [OSSC.Model.SoundItem](#)  
*Used by [CategoryItem](#) to store sounds data.*
- class [OSSC.Model.CustomRange](#)  
*Used by [SoundItem](#) to store Random Ranges.*

#### Namespaces

- namespace [OSSC.Model](#)

### 3.16 SoundItem.cs

```

00001 using UnityEngine;
00002
00003 namespace OSSC.Model
00004 {
00008     [System.Serializable]
00009     public class SoundItem
00010     {
00014         public string name;
00018         public int tagID = -1;
00022         public UnityEngine.Audio.AudioMixerGroup mixer;
00026         public AudioClip[] clips;
00030         public bool isRandomPitch;
00034         public CustomRange pitchRange = new CustomRange();
00038         public bool isRandomVolume;
00042         public CustomRange volumeRange = new CustomRange();
00043
00047         [RangeAttribute(0f, 1f)]
00048         public float volume = 1f;
00049     }
00050
00054     [System.Serializable]
00055     public class CustomRange
00056     {
00060         public float min = 1f;
00064         public float max = 1f;
00065
00070         public float GetRandomRange()
00071         {
00072             return Random.Range(min, max);
00073         }
00074     }
00075 }

```

### 3.17 F:/git-projects/audio-controller-unity/AudioController/Assets/OSSC/Source/Object↔ Pool.cs File Reference

#### Classes

- class [ObjectPool](#)  
*Creates a pool of different prefabs when someone requests a GameObject.*
- class [PrefabBasedPool](#)

## 3.18 ObjectPool.cs

```

00001 using System.Collections;
00002 using System.Collections.Generic;
00003 using UnityEngine;
00004
00008 public class ObjectPool : MonoBehaviour
00009 {
00010     #region Public fields
00011     public List<PrefabBasedPool> pools;
00015     #endregion
00016     #region Public methods and properties
00017
00023     public GameObject GetFreeObject(GameObject prefab = null)
00024     {
00025         if (prefab == null)
00026             return null;
00027
00028         PrefabBasedPool pool = pools.Find((x) => {
00029             return x.prefab == prefab;
00030         });
00031
00032         if (pool != null)
00033             return pool.GetFreeObject();
00034
00035         pool = new PrefabBasedPool(prefab);
00036         GameObject parent = new GameObject();
00037         parent.name = pool.prefab.name + " :: POOL";
00038         parent.transform.parent = this.gameObject.transform;
00039         pool.parent = parent.transform;
00040         pools.Add(pool);
00041         return pool.GetFreeObject();
00042     }
00043
00044     #endregion
00045
00046     #region MonoBehaviour methods
00047     void Awake()
00051     {
00052         pools = new List<PrefabBasedPool>();
00053     }
00054     #endregion
00055 }
00056
00057 [System.Serializable]
00058 public class PrefabBasedPool
00059 {
00060     public PrefabBasedPool(GameObject prefab)
00061     {
00062         pool = new List<GameObject>();
00063         this.prefab = prefab;
00064     }
00065     public GameObject prefab;
00066     public List<GameObject> pool;
00067
00071     public Transform parent;
00072
00073     public GameObject GetFreeObject()
00074     {
00075         GameObject freeObj = pool.Find((x) => {
00076             if (x == null)
00077             {
00078                 pool.Remove(x);
00079                 return false;
00080             }
00081             var poolable = x.GetComponent<IPoolable>();
00082             return poolable.IsFree();
00083         });
00084
00085         if (freeObj != null)
00086         {
00087             freeObj.SetActive(true);
00088             return freeObj;
00089         }
00090
00091         var obj = GameObject.Instantiate(prefab, Vector3.zero, Quaternion.identity, parent);
00092         obj.SetActive(true);
00093         var objPoolable = obj.GetComponent<IPoolable>();
00094         objPoolable.pool = this;
00095         pool.Add(obj);
00096
00097         return obj;
00098     }
00099
00100     public void Despawn(GameObject obj)
00101     {

```

```

00102         obj.transform.SetParent(parent, false);
00103         obj.SetActive(false);
00104     }
00105 }

```

### 3.19 F:/git-projects/audio-controller-unity/AudioController/Assets/OSSC/Source/SoundController.cs File Reference

#### Classes

- class [OSSC.SoundController](#)  
*The main class that is used for Playing and controlling all sounds.*
- struct [OSSC.PlaySoundSettings](#)  
*Set the settings to play a particular cue with particular preferences.*

#### Namespaces

- namespace [OSSC](#)

### 3.20 SoundController.cs

```

00001 using System.Collections;
00002 using System.Collections.Generic;
00003 using UnityEngine;
00004 using OSSC.Model;
00005
00006 namespace OSSC
00007 {
00011     [RequireComponent(typeof(ObjectPool))]
00012     public class SoundController : MonoBehaviour
00013     {
00014         #region Serialized Data
00015         public GameObject _defaultPrefab;
00023         public SoundControllerData _database;
00024
00025         #endregion
00026
00027         #region Private fields
00028
00032         private ObjectPool _pool;
00036         private CueManager _cueManager;
00040         private int _initialCueManagerSize = 10;
00041
00042         #endregion
00043
00044         #region Public methods and properties
00045
00049         public GameObject defaultPrefab
00050         {
00051             set
00052             {
00053                 _defaultPrefab = value;
00054             }
00055         }
00056
00061         public void StopAll(bool shouldCallOnEndCallback = true)
00062         {
00063             _cueManager.StopAllCues(shouldCallOnEndCallback);
00064         }
00065
00071         public void SetMute(string categoryName, bool value)
00072         {
00073             for (int i = 0; i < _database.items.Length; i++)
00074             {
00075                 if (_database.items[i].name == categoryName)
00076                 {
00077                     _database.items[i].isMute = value;

```

```

00078         }
00079     }
00080 }
00081
00082 public ISoundCue Play(PlaySoundSettings settings)
00083 {
00084     if (settings.soundCueProxy != null)
00085     {
00086         return PlaySoundCue(settings);
00087     }
00088
00089     if (settings.names == null && string.IsNullOrEmpty(settings.
name))
00090     {
00091         return null;
00092     }
00093
00094     string[] names = null;
00095     string categoryName = settings.categoryName;
00096     float fadeInTime = settings.fadeInTime;
00097     float fadeOutTime = settings.fadeOutTime;
00098     bool isLooped = settings.isLooped;
00099     int tagID = _database.soundTags.GetTagIDByName(settings.
tagName);
00100
00101     Transform parent = settings.parent;
00102
00103     if (settings.names != null)
00104     {
00105         names = settings.names;
00106     }
00107     else
00108     {
00109         names = new[] { settings.name };
00110     }
00111
00112     UnityEngine.Assertions.Assert.IsNotNull(names, "[AudioController] names cannot be
null");
00113
00114     if (names != null)
00115     {
00116         UnityEngine.Assertions.Assert.IsFalse(names.Length == 0, "[AudioController]
names cannot have 0 strings");
00117
00118         CategoryItem category = null;
00119         GameObject prefab = null;
00120         List<SoundItem> items = new List<SoundItem>();
00121         List<float> catVolumes = new List<float>();
00122         List<CategoryItem> categories = new List<CategoryItem>();
00123
00124         if (string.IsNullOrEmpty(categoryName) == false)
00125         {
00126             category = System.Array.Find(_database.items, (item) =>
00127             {
00128                 return item.name == categoryName;
00129             });
00130
00131             // Debug.Log(category);
00132             if (category == null)
00133                 return null;
00134
00135             prefab = category.usingDefaultPrefab ?
_defaultPrefab : category.audioObjectPrefab;
00136
00137             for (int i = 0; i < names.Length; i++)
00138             {
00139                 SoundItem item = System.Array.Find(category.
soundItems, (x) =>
00140                 {
00141                     return x.name == names[i];
00142                 });
00143
00144                 if (item != null && category.isMute == false)
00145                 {
00146                     bool canAddItem = tagID == -1 || tagID == item.tagID;
00147                     if (canAddItem)
00148                     {
00149                         catVolumes.Add(category.categoryVolume);
00150                         items.Add(item);
00151                         categories.Add(category);
00152                     }
00153                 }
00154             }
00155         }
00156     }
00157     else
00158     {
00159         prefab = _defaultPrefab;
00160         CategoryItem[] categoryItems = _database.items;
00161         for (int i = 0; i < names.Length; i++)
00162         {
00163             SoundItem item = null;
00164             item = items.Find((x) => names[i] == x.name);

```

```

00164         if (item != null)
00165         {
00166             bool canAddItem = tagID == -1 || tagID == item.tagID;
00167             if (canAddItem == false)
00168                 continue;
00169
00170             catVolumes.Add(catVolumes[items.IndexOf(item)]);
00171             categories.Add(categories[items.IndexOf(item)]);
00172             items.Add(item);
00173             continue;
00174         }
00175
00176         for (int j = 0; j < categoryItems.Length; j++)
00177         {
00178             item = System.Array.Find(categoryItems[j].soundItems, (x) => x.
name == names[i]);
00179             if (item != null && categoryItems[j].isMute == false)
00180             {
00181                 bool canAddItem = tagID == -1 || tagID == item.tagID;
00182                 if (canAddItem == false)
00183                     continue;
00184                 catVolumes.Add(categoryItems[j].categoryVolume);
00185                 categories.Add(categoryItems[j]);
00186                 items.Add(item);
00187                 break;
00188             }
00189         }
00190     }
00191 }
00192
00193 if (items.Count == 0)
00194     return null;
00195
00196 SoundCue cue = _cueManager.GetSoundCue();
00197 SoundCueData data;
00198 data.audioPrefab = prefab;
00199 data.sounds = items.ToArray();
00200 data.categoryVolumes = catVolumes.ToArray();
00201 data.categoriesForSounds = categories.ToArray();
00202 data.fadeInTime = fadeInTime;
00203 data.fadeOutTime = fadeOutTime;
00204 data.isFadeIn = data.fadeInTime >= 0.1f;
00205 data.isFadeOut = data.fadeOutTime >= 0.1f;
00206 data.isLooped = isLooped;
00207 cue.AudioObject = _pool.GetFreeObject(prefab).GetComponent<
SoundObject>();
00208 if (parent != null)
00209     cue.AudioObject.transform.SetParent(parent, false);
00210
00211 SoundCueProxy proxy = new SoundCueProxy();
00212 proxy.SoundCue = cue;
00213 proxy.Play(data);
00214 return proxy;
00215 }
00216
00217 #endregion
00218
00219 #region Private methods
00220 private SoundCueProxy PlaySoundCue(PlaySoundSettings settings)
00221 {
00222     SoundCueProxy cue = settings.soundCueProxy as
SoundCueProxy;
00223     Transform parent = settings.parent;
00224     float fadeInTime = settings.fadeInTime;
00225     float fadeOutTime = settings.fadeOutTime;
00226     bool isLooped = settings.isLooped;
00227     var ncue = _cueManager.GetSoundCue();
00228     ncue.AudioObject = _pool.GetFreeObject(cue.
Data.audioPrefab).GetComponent<SoundObject>();
00229     if (parent != null)
00230         ncue.AudioObject.transform.SetParent(parent, false);
00231     SoundCueData data = cue.Data;
00232     data.fadeInTime = fadeInTime;
00233     data.fadeOutTime = fadeOutTime;
00234     data.isFadeIn = data.fadeInTime >= 0.1f;
00235     data.isFadeOut = data.fadeOutTime >= 0.1f;
00236     data.isLooped = isLooped;
00237     cue.SoundCue = ncue;
00238     cue.Play(data);
00239     return cue;
00240 }
00241
00242 #region Internal tests
00243 [ContextMenu("Test play")]
00244 void Test()
00245 {
00246     PlaySoundSettings settings = new PlaySoundSettings();

```



```

00253         settings.Init();
00254         settings.name = "Test";
00255         var proxyCue = Play(settings);
00256         Debug.Log(proxyCue.ID);
00257     }
00258
00259     [ContextMenu("Test Play looped")]
00260     void TestLoop()
00261     {
00262         PlaySoundSettings settings = new PlaySoundSettings();
00263         settings.Init();
00264         settings.name = "Test";
00265         settings.isLooped = true;
00266         var proxyCue = Play(settings);
00267         Debug.Log(proxyCue.ID);
00268     }
00269
00270     [ContextMenu("Test sequence")]
00271     void TestSequence()
00272     {
00273         PlaySoundSettings settings = new PlaySoundSettings();
00274         settings.Init();
00275         settings.names = new[] {"Test", "Test1", "Test2"};
00276         var proxyCue = Play(settings);
00277         Debug.Log(proxyCue.ID);
00278     }
00279
00280     [ContextMenu("Test sequence looped")]
00281     void TestSequenceLooped()
00282     {
00283         PlaySoundSettings settings = new PlaySoundSettings();
00284         settings.Init();
00285         settings.names = new[] {"Test", "Test1", "Test2"};
00286         settings.isLooped = true;
00287         var proxyCue = Play(settings);
00288         Debug.Log(proxyCue.ID);
00289     }
00290
00291     [ContextMenu("Test sequence plays 2 times")]
00292     void TestSequence2TimesPlay()
00293     {
00294         PlaySoundSettings settings = new PlaySoundSettings();
00295         settings.Init();
00296         settings.names = new[] {"Test", "Test1", "Test2"};
00297         var proxyCue = Play(settings);
00298         proxyCue.OnPlayCueEnded += cue =>
00299         {
00300             var sett = new PlaySoundSettings();
00301             sett.soundCueProxy = proxyCue;
00302             proxyCue = Play(sett);
00303         };
00304         Debug.Log(proxyCue.ID);
00305     }
00306     #endregion
00307
00308     #endregion
00309
00310     #region MonoBehaviour methods
00311
00312     void Awake()
00313     {
00314         _pool = GetComponent<ObjectPool>();
00315         _cueManager = new CueManager(_initialCueManagerSize);
00316     }
00317
00318     #endregion
00319 }
00320
00324 public struct PlaySoundSettings
00325 {
00326     public string name;
00327     public string[] names;
00328     public Transform parent;
00329     public float fadeInTime;
00330     public float fadeOutTime;
00331     public string categoryName;
00332     public bool isLooped;
00333     public ISoundCue soundCueProxy;
00334     public string tagName;
00335
00336     public void Init()
00337     {
00338         name = string.Empty;
00339         names = null;
00340         parent = null;
00341         fadeInTime = 0f;
00342         fadeOutTime = 0f;
00343     }
00344 }

```

```

00375         categoryName = string.Empty;
00376         isLooped = false;
00377         soundCueProxy = null;
00378         tagName = string.Empty;
00379     }
00380 }
00381
00382 }

```

## 3.21 F:/git-projects/audio-controller-unity/AudioController/Assets/OSSC/Source/SoundCue.cs File Reference ↩↪

### Classes

- class [OSSC.SoundCue](#)  
*Plays a whole cue of soundItems*
- struct [OSSC.SoundCueData](#)  
*Used for sending data to play to AudioCue*

### Namespaces

- namespace [OSSC](#)

## 3.22 SoundCue.cs

```

00001 using System.Collections;
00002 using System.Collections.Generic;
00003 using UnityEngine;
00004 using System;
00005 using OSSC.Model;
00006
00007 namespace OSSC
00008 {
00012     public class SoundCue : ISoundCue
00013     {
00017         public Action<string> OnPlayEnded { get; set; }
00018
00022         public Action<SoundCue> OnPlayCueEnded { get; set; }
00023
00027         public Action<SoundCue, SoundCueProxy> OnPlayKilled { get; set; }
00028
00032         public SoundObject AudioObject { get; set; }
00033
00037         public SoundCueData Data { get { return _data; } }
00038
00042         public bool IsPlaying
00043         {
00044             get;
00045             private set;
00046         }
00047
00052         public int ID
00053         {
00054             get;
00055             private set;
00056         }
00057
00061         public SoundCue()
00062         {
00063         }
00064
00069         public SoundCue(int id)
00070         {
00071             ID = id;
00072         }
00073
00077         private int _currentItem = 0;

```

```

00081     private SoundCueData _data;
00085     private SoundCueProxy _currentProxy;
00086
00091     public void Play(SoundCueData data)
00092     {
00093         _data = data;
00094         AudioObject.isDespawnOnFinishedPlaying = !data.
isLooped;
00095         AudioObject.OnFinishedPlaying = OnFinishedPlaying_handler;
00096         // audioObject.isDespawnOnFinishedPlaying = false;
00097         if (TryPlayNext() == false)
00098         {
00099             return;
00100         }
00101         IsPlaying = true;
00102     }
00103
00109     public void Play(SoundCueData data, SoundCueProxy proxy)
00110     {
00111         Play(data);
00112         _currentProxy = proxy;
00113     }
00114
00118     public void Pause()
00119     {
00120         UnityEngine.Assertions.Assert.IsTrue(_currentItem > 0, "[AudioCue] Cannot pause when
not even started.");
00121         AudioObject.Pause();
00122     }
00123
00127     public void Resume()
00128     {
00129         UnityEngine.Assertions.Assert.IsTrue(_currentItem > 0, "[AudioCue] Cannot resume
when not even started.");
00130         AudioObject.Resume();
00131     }
00132
00137     public void Stop(bool shouldCallOnFinishedCue = true)
00138     {
00139         if (IsPlaying == false)
00140             return;
00141         AudioObject.OnFinishedPlaying = null;
00142         // ((IPoolable)audioObject).pool.Despawn(audioObject.gameObject);
00143         AudioObject.Stop();
00144         AudioObject = null;
00145         _currentItem = 0;
00146         IsPlaying = false;
00147
00148         if (shouldCallOnFinishedCue)
00149         {
00150             if (OnPlayCueEnded != null)
00151             {
00152                 OnPlayCueEnded(this);
00153             }
00154         }
00155
00156         if (OnPlayKilled != null)
00157         {
00158             OnPlayKilled(this, _currentProxy);
00159             _currentProxy = null;
00160         }
00161     }
00162
00167     private void OnFinishedPlaying_handler(SoundObject obj)
00168     {
00169         string itemName = _data.sounds[_currentItem - 1].name;
00170         if (OnPlayEnded != null) {
00171             OnPlayEnded(itemName);
00172         }
00173
00174         if (_currentItem < _data.sounds.Length)
00175         {
00176             if (TryPlayNext() == false)
00177             {
00178                 Stop(true);
00179             }
00180         }
00181         else
00182         {
00183             if (_data.isLooped)
00184             {
00185                 _currentItem = 0;
00186                 if (TryPlayNext() == false)
00187                 {
00188                     Stop(true);
00189                 }
00190             }
00191         }

```

```

00191         else
00192         {
00193             Stop(true);
00194         }
00195     }
00196 }
00197
00202 private bool TryPlayNext()
00203 {
00204     bool isPlaying = false;
00205     if (_data.categoriesForSounds[_currentItem].
isMute == false)
00206     {
00207         PlayCurrentItem();
00208         _currentItem += 1;
00209         isPlaying = true;
00210     }
00211     else
00212     {
00213         for (int i = _currentItem; i < _data.sounds.Length; i++)
00214         {
00215             if (_data.categoriesForSounds[i].isMute == false)
00216             {
00217                 _currentItem = i;
00218                 PlayCurrentItem();
00219                 _currentItem += 1;
00220                 isPlaying = true;
00221                 break;
00222             }
00223         }
00224     }
00225     return isPlaying;
00226 }
00227
00231 private void PlayCurrentItem()
00232 {
00233     SoundItem item = _data.sounds[_currentItem];
00234
00235     float itemVolume = item.isRandomVolume
00236         ? item.volumeRange.GetRandomRange()
00237         : item.volume;
00238     float realVolume = itemVolume * _data.categoryVolumes[_currentItem];
00239
00240     float realPitch = item.isRandomPitch
00241         ? item.pitchRange.GetRandomRange()
00242         : 1f;
00243
00244     if (_currentItem == _data.sounds.Length - 1)
00245     {
00246         AudioObject.Setup(
00247             item.name,
00248             GetRandomClip( item.clips ),
00249             realVolume,
00250             _data.fadeInTime,
00251             _data.fadeOutTime,
00252             item.mixer,
00253             realPitch);
00254     }
00255     else
00256     {
00257         AudioObject.Setup(
00258             item.name,
00259             GetRandomClip( item.clips ),
00260             realVolume,
00261             mixer: item.mixer,
00262             pitch: realPitch);
00263     }
00264     AudioObject.Play();
00265 }
00266
00272 private AudioClip GetRandomClip(AudioClip[] clips)
00273 {
00274     int index = UnityEngine.Random.Range(0, clips.Length);
00275     return clips[index];
00276 }
00277 }
00278
00282 public struct SoundCueData
00283 {
00287     public SoundItem[] sounds;
00291     public CategoryItem[] categoriesForSounds;
00295     public float[] categoryVolumes;
00299     public GameObject audioPrefab;
00303     public float fadeInTime;
00307     public float fadeOutTime;
00311     public bool isFadeIn;
00315     public bool isFadeOut;

```

```

00316
00320         public bool isLooped;
00321     }
00322 }

```

### 3.23 F:/git-projects/audio-controller-unity/AudioController/Assets/OSSC/Source/SoundCueProxy.cs File Reference

#### Classes

- class [OSSC.SoundCueProxy](#)

Given by the [SoundController](#) to User as [ISoundCue](#) to control the playing [SoundCue](#).

#### Namespaces

- namespace [OSSC](#)

### 3.24 SoundCueProxy.cs

```

00001 using System;
00002 using System.Collections;
00003 using System.Collections.Generic;
00004 using UnityEngine;
00005
00006 namespace OSSC
00007 {
00011     public class SoundCueProxy : ISoundCue
00012     {
00013         #region Private fields
00014         private SoundCue _soundCue;
00021         private SoundCueData _data;
00022         #endregion
00023         #region Public Methods and Properties
00024         public SoundCue SoundCue
00028         {
00029             get {
00030                 return _soundCue;
00031             }
00032             set {
00033                 _soundCue = value;
00034             }
00035         }
00036         #endregion
00037
00038         #region ISoundCue implementation
00039         public Action<string> OnPlayEnded
00043         {
00044             get {
00045                 return _soundCue == null ? null : _soundCue.OnPlayEnded;
00046             }
00047             set {
00048                 if (_soundCue != null)
00049                 {
00050                     _soundCue.OnPlayEnded = value;
00051                 }
00052             }
00053         }
00057         public Action<SoundCue> OnPlayCueEnded
00058         {
00059             get {
00060                 return _soundCue == null ? null : _soundCue.OnPlayCueEnded;
00061             }
00062             set {
00063                 if (_soundCue != null)
00064                 {
00065                     _soundCue.OnPlayCueEnded = value;
00066                 }
00067             }
00068         }
00069     }
00070 }

```

```

00068     }
00072     public SoundObject AudioObject
00073     {
00074         get {
00075             return _soundCue == null ? null : _soundCue.AudioObject;
00076         }
00077         set {
00078             if (_soundCue != null)
00079             {
00080                 _soundCue.AudioObject = value;
00081             }
00082         }
00083     }
00084
00088     public SoundCueData Data { get { return _data; } }
00089
00093     public bool IsPlaying
00094     {
00095         get {
00096             return _soundCue != null && _soundCue.IsPlaying;
00097         }
00098     }
00099
00103     public int ID
00104     {
00105         get {
00106             return _soundCue == null ? -999 : _soundCue.ID;
00107         }
00108     }
00109
00113     public void Play(SoundCueData data)
00114     {
00115         _data = data;
00116         if (_soundCue == null)
00117         {
00118             Debug.LogError("NO SOUND CUE to play!!!");
00119             return;
00120         }
00121         _soundCue.Play(data, this);
00122     }
00123
00127     public void Pause()
00128     {
00129         if (_soundCue == null)
00130         {
00131             Debug.LogError("NO SOUND CUE to pause!!!");
00132             return;
00133         }
00134         _soundCue.Pause();
00135     }
00136
00140     public void Resume()
00141     {
00142         if (_soundCue == null)
00143         {
00144             Debug.LogError("NO SOUND CUE to Resume!!!");
00145             return;
00146         }
00147         _soundCue.Resume();
00148     }
00149
00153     public void Stop(bool shouldCallOnFinishedCue = true)
00154     {
00155         if (_soundCue == null)
00156         {
00157             Debug.LogError("NO SOUND CUE to Stop!!!");
00158             return;
00159         }
00160         _soundCue.Stop(shouldCallOnFinishedCue);
00161     }
00162     #endregion
00163 }
00164 }

```

### 3.25 F:/git-projects/audio-controller-unity/AudioController/Assets/OSSC/Source/SoundObject.cs File Reference ↩

#### Classes

- class [OSSC.SoundObject](#)

Used by the [SoundCue](#). Controls the AudioSource.

## Namespaces

- namespace [OSSC](#)

## 3.26 SoundObject.cs

```

00001 using System;
00002 using System.Collections;
00003 using System.Collections.Generic;
00004 using UnityEngine;
00005 using UnityEngine.Audio;
00006
00007 namespace OSSC
00008 {
00013     [RequireComponent(typeof(AudioSource))]
00014     public class SoundObject : MonoBehaviour, IPoolable
00015     {
00016
00020         public System.Action<SoundObject> OnFinishedPlaying;
00021
00022         #region private fields
00023         private string _id;
00030         private AudioClip _clip;
00034         private AudioSource _source;
00035
00039         private Coroutine _playingRoutine;
00043         private bool _isPaused;
00044
00048         private bool _isFree = true;
00052         private PrefabBasedPool _pool;
00056         private float _fadeInTime;
00060         private float _fadeOutTime;
00064         private float _volume;
00068         private float _pitch;
00069         private bool _isDespawnOnFinishedPlaying = true;
00070         #endregion
00071
00072         #region Public methods and properties
00073         public bool isDespawnOnFinishedPlaying {
00077             get { return _isDespawnOnFinishedPlaying; }
00078             set { _isDespawnOnFinishedPlaying = value; }
00079         }
00080
00084         public string clipName
00085         {
00086             get
00087             {
00088                 return _clip != null ? _clip.name : "NONE";
00089             }
00090         }
00091
00095         public AudioSource source
00096         {
00097             get { return _source; }
00098         }
00099
00103         public string ID {
00104             get { return _id; }
00105         }
00106
00117         public void Setup(string id, AudioClip clip, float volume, float fadeInTime = 0f, float
fadeOutTime = 0f, AudioManager mixer = null, float pitch = 1f)
00118         {
00119             _id = id;
00120             _clip = clip;
00121             gameObject.name = _id;
00122             if (_source == null)
00123                 _source = GetComponent<AudioSource>();
00124
00125             _source.volume = 0;
00126             _source.time = 0f;
00127             _source.outputAudioMixerGroup = mixer;
00128             _volume = volume;
00129             _pitch = pitch;
00130             _fadeInTime = fadeInTime;
00131             _fadeOutTime = fadeOutTime;

```

```

00132     }
00133
00137 public void Play()
00138 {
00139     if (_source == null)
00140         _source = GetComponent<AudioSource>();
00141     _source.clip = _clip;
00142     gameObject.SetActive(true);
00143     _source.pitch = _pitch;
00144     StartCoroutine(FadeRoutine(_fadeInTime, _volume));
00145     _source.Play();
00146     _isFree = false;
00147     _playingRoutine = StartCoroutine(PlayingRoutine());
00148 }
00149
00153 public void Pause()
00154 {
00155     if (_source == null)
00156         return;
00157     _isPaused = true;
00158     _source.Pause();
00159 }
00160
00164 public void Resume()
00165 {
00166     if (_source == null)
00167         return;
00168     _source.Play();
00169     _isPaused = false;
00170 }
00171
00175 public void Stop()
00176 {
00177     if (_playingRoutine == null)
00178         return;
00179
00180     StartCoroutine(StopRoutine());
00181 }
00182
00186 [ContextMenu("Test Play")]
00187 private void TestPlay()
00188 {
00189     Play();
00190 }
00191 #endregion
00192
00199 private IEnumerator FadeRoutine(float fadeTime, float value)
00200 {
00201     if (fadeTime < 0.1f)
00202     {
00203         _source.volume = value;
00204         yield break;
00205     }
00206
00207     float initVal = _source.volume;
00208     float fadeSpeed = 1f / (fadeTime / Time.deltaTime);
00209     for (float t = 0f; t < 1f; t += fadeSpeed)
00210     {
00211         float val = Mathf.SmoothStep(initVal, value, t);
00212         _source.volume = val;
00213         yield return null;
00214     }
00215
00216     _source.volume = value;
00217 }
00218
00223 private IEnumerator StopRoutine()
00224 {
00225     StopCoroutine(_playingRoutine);
00226     yield return StartCoroutine(FadeRoutine(_fadeOutTime, 0f));
00227     _source.Stop();
00228     _source.clip = null;
00229     _playingRoutine = null;
00230     _isFree = true;
00231     _volume = 0f;
00232     _source.time = 0f;
00233     _source.pitch = 1f;
00234
00235     if (isDespawnOnFinishedPlaying)
00236         _pool.Despawn(gameObject);
00237
00238     if (OnFinishedPlaying != null)
00239     {
00240         OnFinishedPlaying(this);
00241     }
00242 }
00243

```



```

00248     private IEnumerator PlayingRoutine()
00249     {
00250         while (true)
00251         {
00252             yield return null;
00253             float fadeOutTrigger = _source.clip.length - _fadeOutTime;
00254             if (_source.time >= fadeOutTrigger)
00255             {
00256                 yield return StartCoroutine(FadeRoutine(_fadeOutTime, 0f));
00257             }
00258             if (!_source.isPlaying && !_isPaused)
00259             {
00260                 break;
00261             }
00262         }
00263
00264         _source.clip = null;
00265         _playingRoutine = null;
00266         _isFree = true;
00267         _volume = 0f;
00268         _source.time = 0f;
00269
00270         if (isDespawnOnFinishedPlaying)
00271             _pool.Despawn(gameObject);
00272
00273         if (OnFinishedPlaying != null)
00274         {
00275             OnFinishedPlaying(this);
00276         }
00277     }
00278
00279     #region IPoolable methods
00280     PrefabBasedPool IPoolable.pool {
00281         get { return _pool; }
00282         set { _pool = value; }
00283     }
00284
00285     public bool IsFree()
00286     {
00287         return _isFree;
00288     }
00289     #endregion
00290 }
00291
00292 }
```

## 3.27 F:/git-projects/audio-controller-unity/AudioController/Assets/OSSC/Source/SoundTags.cs File Reference

### Classes

- class [OSSC.SoundTags](#)  
*Used By the [SoundController](#) for tagging SoundItems.*
- struct [OSSC.TagData](#)  
*Used by the [SoundTags](#) to save Tags.*

### Namespaces

- namespace [OSSC](#)

## 3.28 SoundTags.cs

```

00001 using System.Collections.Generic;
00002 using UnityEngine;
00003
00004 namespace OSSC
00005 {
```

```

00009     [System.Serializable]
00010     public class SoundTags
00011     {
00012         #region Private fields
00013         [SerializeField]
00017         private List<TagData> _tagsData;
00018
00019         private int _lastID = 0;
00020         #endregion
00021
00025         public SoundTags()
00026         {
00027             _tagsData = new List<TagData>();
00028         }
00029
00030         #region Public methods and properties
00031
00036         public TagData[] ToArray()
00037         {
00038             return _tagsData.ToArray();
00039         }
00040
00045         public string[] ToArrayNames()
00046         {
00047             string[] names = new string[_tagsData.Count];
00048             for (int i = 0; i < names.Length; i++)
00049             {
00050                 names[i] = _tagsData[i].name;
00051             }
00052
00053             return names;
00054         }
00055
00060         public int[] ToArrayIDs()
00061         {
00062             int[] ids = new int[_tagsData.Count];
00063             for (int i = 0; i < ids.Length; i++)
00064             {
00065                 ids[i] = _tagsData[i].ID;
00066             }
00067
00068             return ids;
00069         }
00070
00076         public TagData GetTagDataByName(string name)
00077         {
00078             return _tagsData.Find(data => data.name.Equals(name.ToLower()));
00079         }
00080
00086         public TagData GetTagDataByID(int ID)
00087         {
00088             return _tagsData.Find(data => data.ID.Equals(ID));
00089         }
00090
00096         public int GetTagIDByName(string name)
00097         {
00098             TagData result = _tagsData.Find(data => data.name.Equals(name.ToLower()));
00099             if (string.IsNullOrEmpty(result.name))
00100                 return -1;
00101
00102             return result.ID;
00103         }
00104
00110         public string GetTagNameByID(int ID)
00111         {
00112             TagData result = _tagsData.Find(data => data.ID.Equals(ID));
00113             if (string.IsNullOrEmpty(result.name))
00114                 return string.Empty;
00115
00116             return result.name;
00117         }
00118
00123         public void SetTag(string name)
00124         {
00125             if (string.IsNullOrEmpty(name))
00126                 return;
00127
00128             string nameLowercase = name.ToLower();
00129             TagData result = _tagsData.Find(data => data.name.Equals(nameLowercase));
00130             if (string.IsNullOrEmpty(result.name) == false)
00131                 return;
00132
00133             result.name = nameLowercase;
00134             result.ID = _lastID;
00135             _lastID += 1;
00136             _tagsData.Add(result);
00137         }

```

```
00138
00143     public void RemoveByTag(TagData data)
00144     {
00145         _tagsData.Remove(data);
00146     }
00147     #endregion
00148 }
00149
00153 [System.Serializable]
00154 public struct TagData
00155 {
00159     public string name;
00163     public int ID;
00164 }
00165 }
```



# Index

- [\\_database](#)
    - [OSSC::SoundController, 21](#)
  - [\\_defaultPrefab](#)
    - [OSSC::SoundController, 21](#)
- [assetName](#)
  - [OSSC::Model::SoundControllerData, 22](#)
- [AudioObject](#)
  - [OSSC::ISoundCue, 12](#)
  - [OSSC::SoundCue, 28](#)
  - [OSSC::SoundCueProxy, 34](#)
- [audioObjectPrefab](#)
  - [OSSC::Model::CategoryItem, 3](#)
- [audioPrefab](#)
  - [OSSC::SoundCueData, 30](#)
- [categoriesForSounds](#)
  - [OSSC::SoundCueData, 31](#)
- [categoryName](#)
  - [OSSC::PlaySoundSettings, 15](#)
- [categoryVolume](#)
  - [OSSC::Model::CategoryItem, 4](#)
- [categoryVolumes](#)
  - [OSSC::SoundCueData, 31](#)
- [clipName](#)
  - [OSSC::SoundObject, 41](#)
- [clips](#)
  - [OSSC::Model::SoundItem, 36](#)
- [CueManager](#)
  - [OSSC::CueManager, 6](#)
- [Data](#)
  - [OSSC::ISoundCue, 12](#)
  - [OSSC::SoundCue, 28](#)
  - [OSSC::SoundCueProxy, 34](#)
- [defaultPrefab](#)
  - [OSSC::SoundController, 21](#)
- [Despawn](#)
  - [PrefabBasedPool, 18](#)
- [F:/git-projects/audio-controller-unity/AudioController/↔](#)
  - [Assets/OSSC/Source/CueManager.cs, 49](#)
- [F:/git-projects/audio-controller-unity/AudioController/↔](#)
  - [Assets/OSSC/Source/Editor/SoundController↔](#)
    - [Editor.cs, 50](#)
- [F:/git-projects/audio-controller-unity/AudioController/↔](#)
  - [Assets/OSSC/Source/Editor/SoundObject↔](#)
    - [Editor.cs, 54, 55](#)
- [F:/git-projects/audio-controller-unity/AudioController/↔](#)
  - [Assets/OSSC/Source/IPoolable.cs, 55](#)
- [F:/git-projects/audio-controller-unity/AudioController/↔](#)
  - [Assets/OSSC/Source/ISoundCue.cs, 56](#)
- [F:/git-projects/audio-controller-unity/AudioController/↔](#)
  - [Assets/OSSC/Source/Model/CategoryItem↔](#)
    - [cs, 56, 57](#)
- [F:/git-projects/audio-controller-unity/AudioController/↔](#)
  - [Assets/OSSC/Source/Model/SoundController↔](#)
    - [Data.cs, 57](#)
- [F:/git-projects/audio-controller-unity/AudioController/↔](#)
  - [Assets/OSSC/Source/Model/SoundItem.cs, 58](#)
- [F:/git-projects/audio-controller-unity/AudioController/↔](#)
  - [Assets/OSSC/Source/ObjectPool.cs, 58, 59](#)
- [F:/git-projects/audio-controller-unity/AudioController/↔](#)
  - [Assets/OSSC/Source/SoundController.cs, 60](#)
- [F:/git-projects/audio-controller-unity/AudioController/↔](#)
  - [Assets/OSSC/Source/SoundCue.cs, 64](#)
- [F:/git-projects/audio-controller-unity/AudioController/↔](#)
  - [Assets/OSSC/Source/SoundCueProxy.cs, 67](#)
- [F:/git-projects/audio-controller-unity/AudioController/↔](#)
  - [Assets/OSSC/Source/SoundObject.cs, 68, 69](#)
- [F:/git-projects/audio-controller-unity/AudioController/↔](#)
  - [Assets/OSSC/Source/SoundTags.cs, 71](#)
- [fadeInTime](#)
  - [OSSC::PlaySoundSettings, 16](#)
  - [OSSC::SoundCueData, 31](#)
- [fadeOutTime](#)
  - [OSSC::PlaySoundSettings, 16](#)
  - [OSSC::SoundCueData, 31](#)
- [foldOutCategories](#)
  - [OSSC::Model::SoundControllerData, 23](#)
- [foldOutSoundItems](#)
  - [OSSC::Model::CategoryItem, 4](#)
- [foldOutTags](#)
  - [OSSC::Model::SoundControllerData, 23](#)
- [GetFreeObject](#)
  - [ObjectPool, 13](#)
  - [PrefabBasedPool, 18](#)
- [GetRandomRange](#)
  - [OSSC::Model::CustomRange, 7](#)
- [GetSoundCue](#)
  - [OSSC::CueManager, 6](#)
- [GetTagDataByID](#)
  - [OSSC::SoundTags, 44](#)
- [GetTagDataByName](#)
  - [OSSC::SoundTags, 44](#)
- [GetTagIDByName](#)
  - [OSSC::SoundTags, 45](#)

- GetTagNameByID
  - OSSC::SoundTags, 45
- IPoolable, 8
  - IsFree, 9
  - pool, 9
- ID
  - OSSC::ISoundCue, 12
  - OSSC::SoundCue, 29
  - OSSC::SoundCueProxy, 35
  - OSSC::SoundObject, 41
  - OSSC::TagData, 47
- Init
  - OSSC::PlaySoundSettings, 15
- isDespawnOnFinishedPlaying
  - OSSC::SoundObject, 42
- isFadeln
  - OSSC::SoundCueData, 31
- isFadeOut
  - OSSC::SoundCueData, 32
- IsFree
  - IPoolable, 9
  - OSSC::SoundObject, 39
- isLooped
  - OSSC::PlaySoundSettings, 16
  - OSSC::SoundCueData, 32
- isMute
  - OSSC::Model::CategoryItem, 4
- IsPlaying
  - OSSC::ISoundCue, 12
  - OSSC::SoundCue, 29
  - OSSC::SoundCueProxy, 35
- isRandomPitch
  - OSSC::Model::SoundItem, 37
- isRandomVolume
  - OSSC::Model::SoundItem, 37
- items
  - OSSC::Model::SoundControllerData, 23
- max
  - OSSC::Model::CustomRange, 8
- min
  - OSSC::Model::CustomRange, 8
- mixer
  - OSSC::Model::SoundItem, 37
- name
  - OSSC::Model::CategoryItem, 4
  - OSSC::Model::SoundItem, 37
  - OSSC::PlaySoundSettings, 16
  - OSSC::TagData, 47
- names
  - OSSC::PlaySoundSettings, 16
- OSSC.CueManager, 5
- OSSC.Editor, 2
- OSSC.Editor.SoundControllerEditor, 24
- OSSC.Editor.SoundObjectEditor, 42
- OSSC.ISoundCue, 10
- OSSC.Model, 2
- OSSC.Model.CategoryItem, 3
- OSSC.Model.CustomRange, 7
- OSSC.Model.SoundControllerData, 22
- OSSC.Model.SoundItem, 36
- OSSC.PlaySoundSettings, 14
- OSSC.SoundController, 19
- OSSC.SoundCue, 24
- OSSC.SoundCueData, 30
- OSSC.SoundCueProxy, 32
- OSSC.SoundObject, 38
- OSSC.SoundTags, 43
- OSSC.TagData, 47
- OSSC::CueManager
  - CueManager, 6
  - GetSoundCue, 6
  - StopAllCues, 6
- OSSC::Editor::SoundControllerEditor
  - OnInspectorGUI, 24
- OSSC::Editor::SoundObjectEditor
  - OnInspectorGUI, 43
- OSSC::ISoundCue
  - AudioObject, 12
  - Data, 12
  - ID, 12
  - IsPlaying, 12
  - OnPlayCueEnded, 12
  - OnPlayEnded, 12
  - Pause, 10
  - Play, 11
  - Resume, 11
  - Stop, 11
- OSSC::Model::CategoryItem
  - audioObjectPrefab, 3
  - categoryVolume, 4
  - foldOutSoundItems, 4
  - isMute, 4
  - name, 4
  - soundItems, 4
  - soundsSearchName, 5
  - usingDefaultPrefab, 5
- OSSC::Model::CustomRange
  - GetRandomRange, 7
  - max, 8
  - min, 8
- OSSC::Model::SoundControllerData
  - assetName, 22
  - foldOutCategories, 23
  - foldOutTags, 23
  - items, 23
  - soundTags, 23
- OSSC::Model::SoundItem
  - clips, 36
  - isRandomPitch, 37
  - isRandomVolume, 37
  - mixer, 37
  - name, 37
  - pitchRange, 37

- tagID, [38](#)
- volume, [38](#)
- volumeRange, [38](#)
- OSSC::PlaySoundSettings
  - categoryName, [15](#)
  - fadeInTime, [16](#)
  - fadeOutTime, [16](#)
  - Init, [15](#)
  - isLooped, [16](#)
  - name, [16](#)
  - names, [16](#)
  - parent, [17](#)
  - soundCueProxy, [17](#)
  - tagName, [17](#)
- OSSC::SoundController
  - \_database, [21](#)
  - \_defaultPrefab, [21](#)
  - defaultPrefab, [21](#)
  - Play, [20](#)
  - SetMute, [20](#)
  - StopAll, [21](#)
- OSSC::SoundCue
  - AudioObject, [28](#)
  - Data, [28](#)
  - ID, [29](#)
  - IsPlaying, [29](#)
  - OnPlayCueEnded, [29](#)
  - OnPlayEnded, [29](#)
  - OnPlayKilled, [29](#)
  - Pause, [26](#)
  - Play, [26](#)
  - Resume, [28](#)
  - SoundCue, [25, 26](#)
  - Stop, [28](#)
- OSSC::SoundCueData
  - audioPrefab, [30](#)
  - categoriesForSounds, [31](#)
  - categoryVolumes, [31](#)
  - fadeInTime, [31](#)
  - fadeOutTime, [31](#)
  - isFadeIn, [31](#)
  - isFadeOut, [32](#)
  - isLooped, [32](#)
  - sounds, [32](#)
- OSSC::SoundCueProxy
  - AudioObject, [34](#)
  - Data, [34](#)
  - ID, [35](#)
  - IsPlaying, [35](#)
  - OnPlayCueEnded, [35](#)
  - OnPlayEnded, [35](#)
  - Pause, [33](#)
  - Play, [33](#)
  - Resume, [34](#)
  - SoundCue, [35](#)
  - Stop, [34](#)
- OSSC::SoundObject
  - clipName, [41](#)
  - ID, [41](#)
  - isDespawnOnFinishedPlaying, [42](#)
  - IsFree, [39](#)
  - OnFinishedPlaying, [41](#)
  - Pause, [40](#)
  - Play, [40](#)
  - Resume, [40](#)
  - Setup, [40](#)
  - source, [42](#)
  - Stop, [41](#)
- OSSC::SoundTags
  - GetTagDataByID, [44](#)
  - GetTagDataByName, [44](#)
  - GetTagIDByName, [45](#)
  - GetTagNameByID, [45](#)
  - RemoveByTag, [45](#)
  - SetTag, [46](#)
  - SoundTags, [44](#)
  - ToArray, [46](#)
  - ToArrayIDs, [46](#)
  - ToArrayNames, [46](#)
- OSSC::TagData
  - ID, [47](#)
  - name, [47](#)
- OSSC, [1](#)
- ObjectPool, [13](#)
  - GetFreeObject, [13](#)
  - pools, [14](#)
- OnFinishedPlaying
  - OSSC::SoundObject, [41](#)
- OnInspectorGUI
  - OSSC::Editor::SoundControllerEditor, [24](#)
  - OSSC::Editor::SoundObjectEditor, [43](#)
- OnPlayCueEnded
  - OSSC::ISoundCue, [12](#)
  - OSSC::SoundCue, [29](#)
  - OSSC::SoundCueProxy, [35](#)
- OnPlayEnded
  - OSSC::ISoundCue, [12](#)
  - OSSC::SoundCue, [29](#)
  - OSSC::SoundCueProxy, [35](#)
- OnPlayKilled
  - OSSC::SoundCue, [29](#)
- parent
  - OSSC::PlaySoundSettings, [17](#)
  - PrefabBasedPool, [18](#)
- Pause
  - OSSC::ISoundCue, [10](#)
  - OSSC::SoundCue, [26](#)
  - OSSC::SoundCueProxy, [33](#)
  - OSSC::SoundObject, [40](#)
- pitchRange
  - OSSC::Model::SoundItem, [37](#)
- Play
  - OSSC::ISoundCue, [11](#)
  - OSSC::SoundController, [20](#)
  - OSSC::SoundCue, [26](#)
  - OSSC::SoundCueProxy, [33](#)

- OSSC::SoundObject, 40
- pool
  - IPoolable, 9
  - PrefabBasedPool, 19
- pools
  - ObjectPool, 14
- prefab
  - PrefabBasedPool, 19
- PrefabBasedPool, 17
  - Despawn, 18
  - GetFreeObject, 18
  - parent, 18
  - pool, 19
  - prefab, 19
  - PrefabBasedPool, 18
- RemoveByTag
  - OSSC::SoundTags, 45
- Resume
  - OSSC::ISoundCue, 11
  - OSSC::SoundCue, 28
  - OSSC::SoundCueProxy, 34
  - OSSC::SoundObject, 40
- SetMute
  - OSSC::SoundController, 20
- SetTag
  - OSSC::SoundTags, 46
- Setup
  - OSSC::SoundObject, 40
- SoundCue
  - OSSC::SoundCue, 25, 26
  - OSSC::SoundCueProxy, 35
- soundCueProxy
  - OSSC::PlaySoundSettings, 17
- soundItems
  - OSSC::Model::CategoryItem, 4
- SoundTags
  - OSSC::SoundTags, 44
- soundTags
  - OSSC::Model::SoundControllerData, 23
- sounds
  - OSSC::SoundCueData, 32
- soundsSearchName
  - OSSC::Model::CategoryItem, 5
- source
  - OSSC::SoundObject, 42
- Stop
  - OSSC::ISoundCue, 11
  - OSSC::SoundCue, 28
  - OSSC::SoundCueProxy, 34
  - OSSC::SoundObject, 41
- StopAll
  - OSSC::SoundController, 21
- StopAllCues
  - OSSC::CueManager, 6
- tagID
  - OSSC::Model::SoundItem, 38
- tagName
  - OSSC::PlaySoundSettings, 17
- ToArray
  - OSSC::SoundTags, 46
- ToArrayIDs
  - OSSC::SoundTags, 46
- ToArrayNames
  - OSSC::SoundTags, 46
- usingDefaultPrefab
  - OSSC::Model::CategoryItem, 5
- volume
  - OSSC::Model::SoundItem, 38
- volumeRange
  - OSSC::Model::SoundItem, 38