AUDIO MANAGER

DOCUMENTATION

Summary

Asset Version	3.1.x or newer
Unity Version	2020.3.x or newer
Price	FREE
Revision	1
Last Updated (Y/M/D)	2025/04/19

The audio manager is designed to do the following:

- Automate an audio library of all the audio clips in your project.
- Provide ways to manage your clips into groups, change their IDs, playback start times, and more.
- Provide a simple API to perform simple playback options for the audio in the library that can be expanded upon.
- Be easy to use, with little to no setup needed from the user to start using the asset.

If you are looking for an asset with super-advanced features and a fully fledged setup for any need possible, this is likely not the asset for you.

The audio manager is built for a more beginner/intermediate level developer. Where most audio setup requirements are simple play clip at x time with a random volume & pitch or similar. With the option to make some extra adjustments.

Now that your expectations are set.
On to how to use the audio manager!

Contents

Installation & Updating

Unity Asset Store

You'll first need to add the asset to your account to access it. If you've already done this the "Add to My Assets" button will display something like "Open In Unity".

Then, in Unity. Access the "Package Manager" via the "Window" tab in the navbar. From there switch to the "My Assets" view for the packages dropdown. From there you'll be able to select the Audio Manager and download/import the latest version. Then all you need to do is import all the files in the package and you'll be good to go.

GitHub (Recommended)

For GitHub you'll need to navigate to the latest release and download the package provided in the assets for the release.

It is recommended to use the repo URL in the package manager in Unity as it keeps the assets' files in a safe space and allows for easy updating. The URL to add it:

https://github.com/CarterGames/AudioManager.git

If you want to use a standard unity package instead, there will always be a .unitypackage file provided in each release. Then you just import that package into your project. You can also just download the source and copy/paste it into your project for a similar result.

Itch.io

For itchio, just download the asset .unitypackage and import it into your project.

Updating The Asset

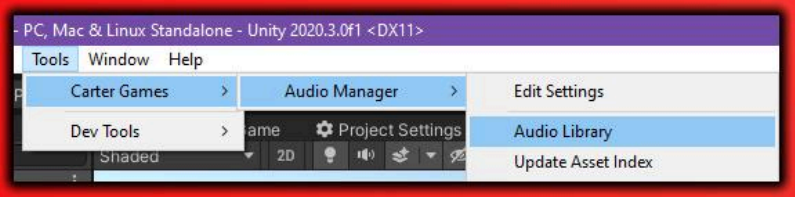
Most updates will be fine to import over an existing install so long as the version you are using is in the same major version. A major version is the first number in the version number. So the current one is **3**.x.x. If that changes, it is recommended to perform a clean install instead. Some updates will require a clean install to function properly. See the release notes for the new version to see if this is necessary.

Clean Install

A clean install is where you remove all files from the asset and then import the new files into the project. It’s recommended when elements break due to updating the asset.

The Audio Library

The audio library is the main GUI you'll be interacting with outside of the settings to manage your audio.

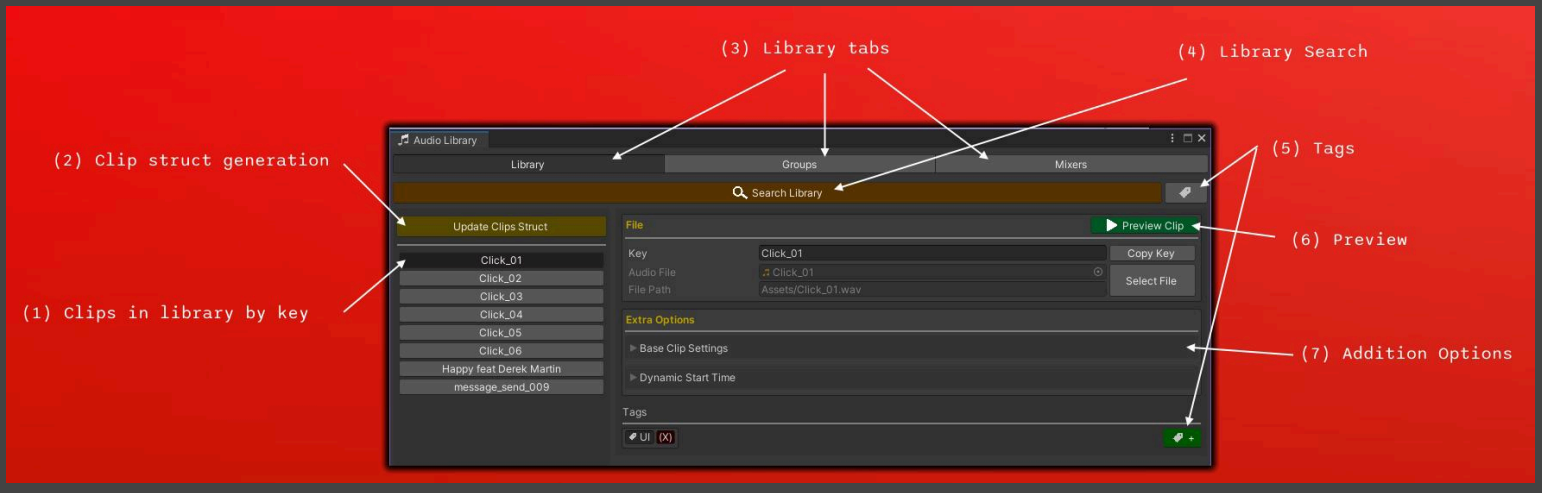


You can access the audio library through the following menu item:

Tools > Carter Games > Audio Manager > Audio Library

Clips

This will open the audio library editor window:



The library tab is where you can view all of your stored audio. This will update when you add or remove clips from your project, once Unity has detected them.

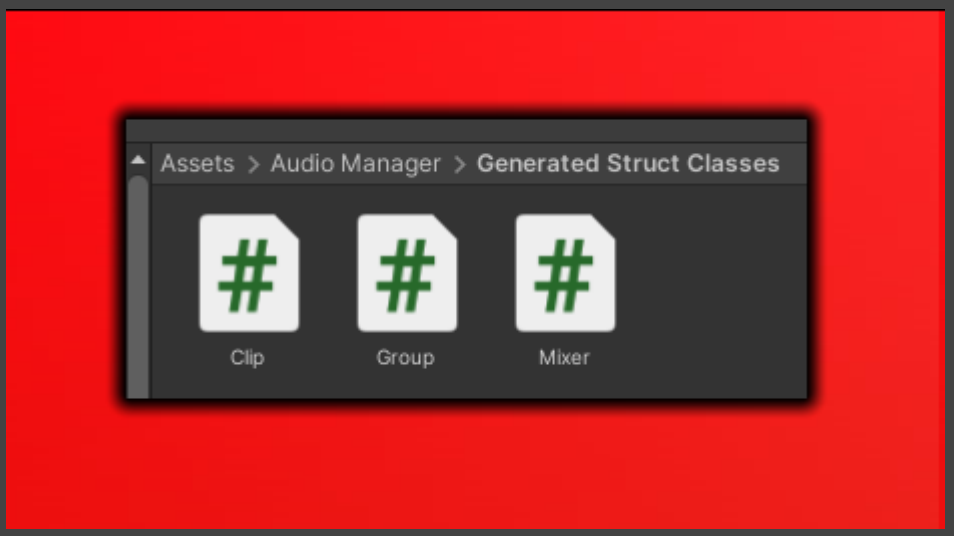
A brief rundown of the GUI:

1. A button is placed on the sidebar for each clip in your project. It will display the key name you have given it. By default, this will be the name of the audio clip when detected. You can edit this value at any time in the library by selecting the clip and editing the key field displayed on the right-hand side.

2. This button will allow you to generate a struct class with constants of all the clip keys. This is useful to avoid typo's when passing in clips to be played with the API. In this example, you would be able to use **Clip.Click_01** to reference this clip in code from the generated class.

   These classes are auto generated for you when you press the button and will be placed in the following pre-defined path. The path will auto-generate if it doesn't already exist:

   Assets / Audio Manager / Generated Struct Classes / ...



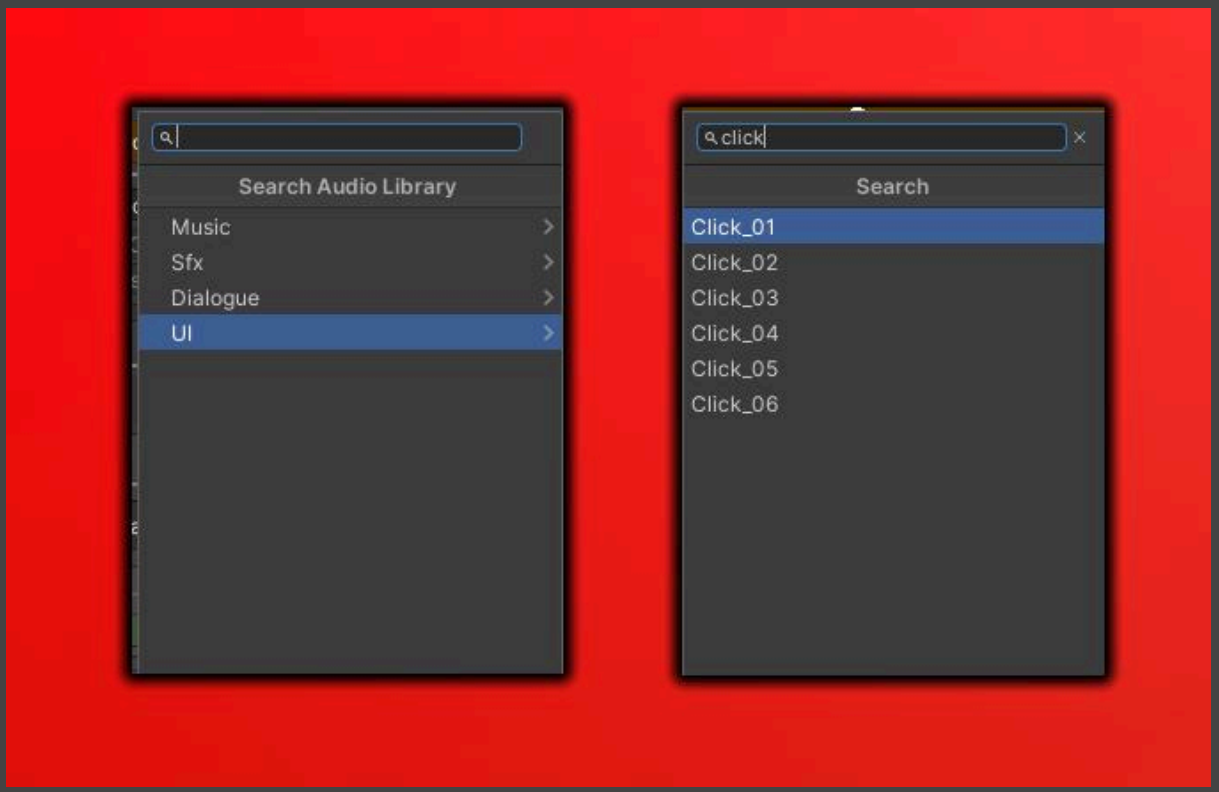
3. The library has tabs for extra settings & options. The other tabs will be explained further in this doc. We will remain in the library tab for now.

4. Pressing this button will open up a search provider with all the clips in the library.

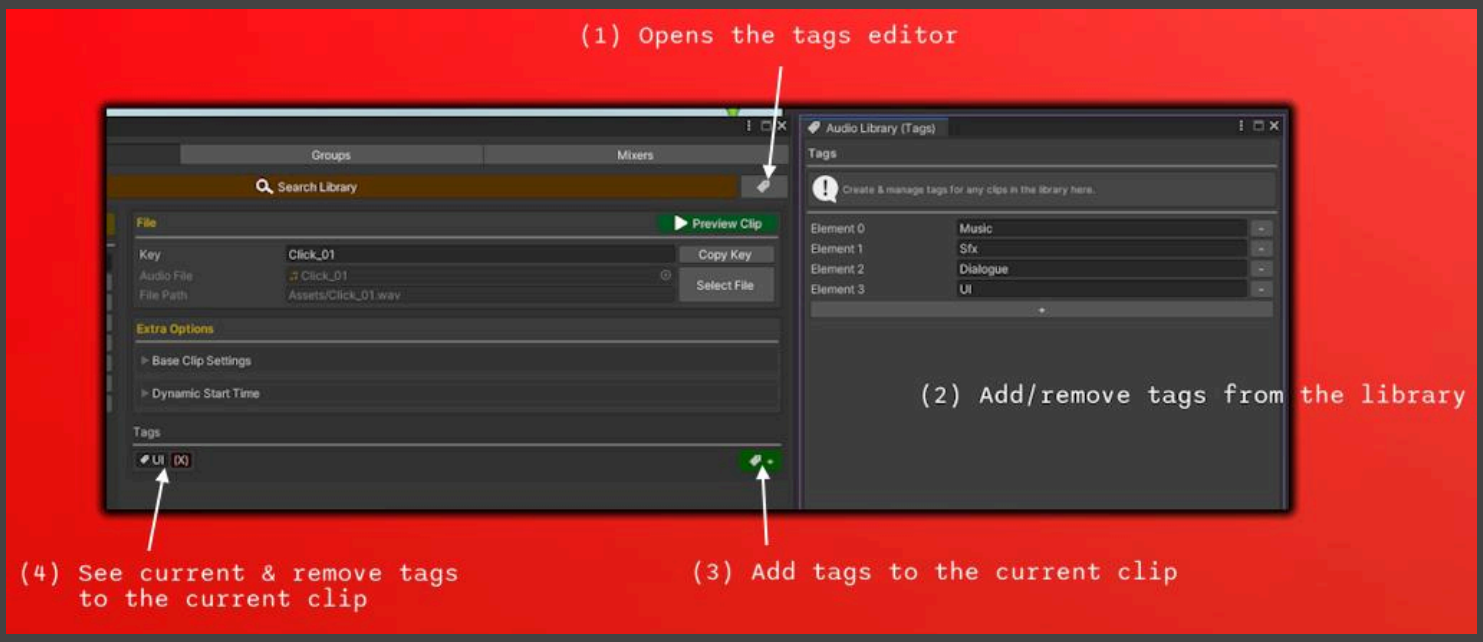
   You can use this to select a specific clip with ease. This is especially useful in projects with a-lot of audio clips stored.

   The categorizing of the clips in the search provider is the tags assigned to each clip.

   Clicking on the entry in the provider will open the clip in question into view so you can see its settings etc.



5. Tags can be used to categorize clips in searches as well as in the API to get a clip by it tags. Just like you can get GameObjects by tag in Unity's API.



5.1.
Opens the tags editor, seen on the right of the image above.

5.2.
You can add or remove tags from the list shown. It's essentially a list of strings, but the source is elsewhere on the audio library asset.

5.3.
Use the button to open to add tags to the current clip with a small search provider GUI.

5.4.
See the current tags on the clip. You can remove a tag by pressing the (x) button next to each entry.

6. Use this button to preview the specific clip in the editor without needing to enter play-mode.

7. Additional settings for each clip can be found here. I'll explain both of these options on the next page.

7.1. Base Clip Settings

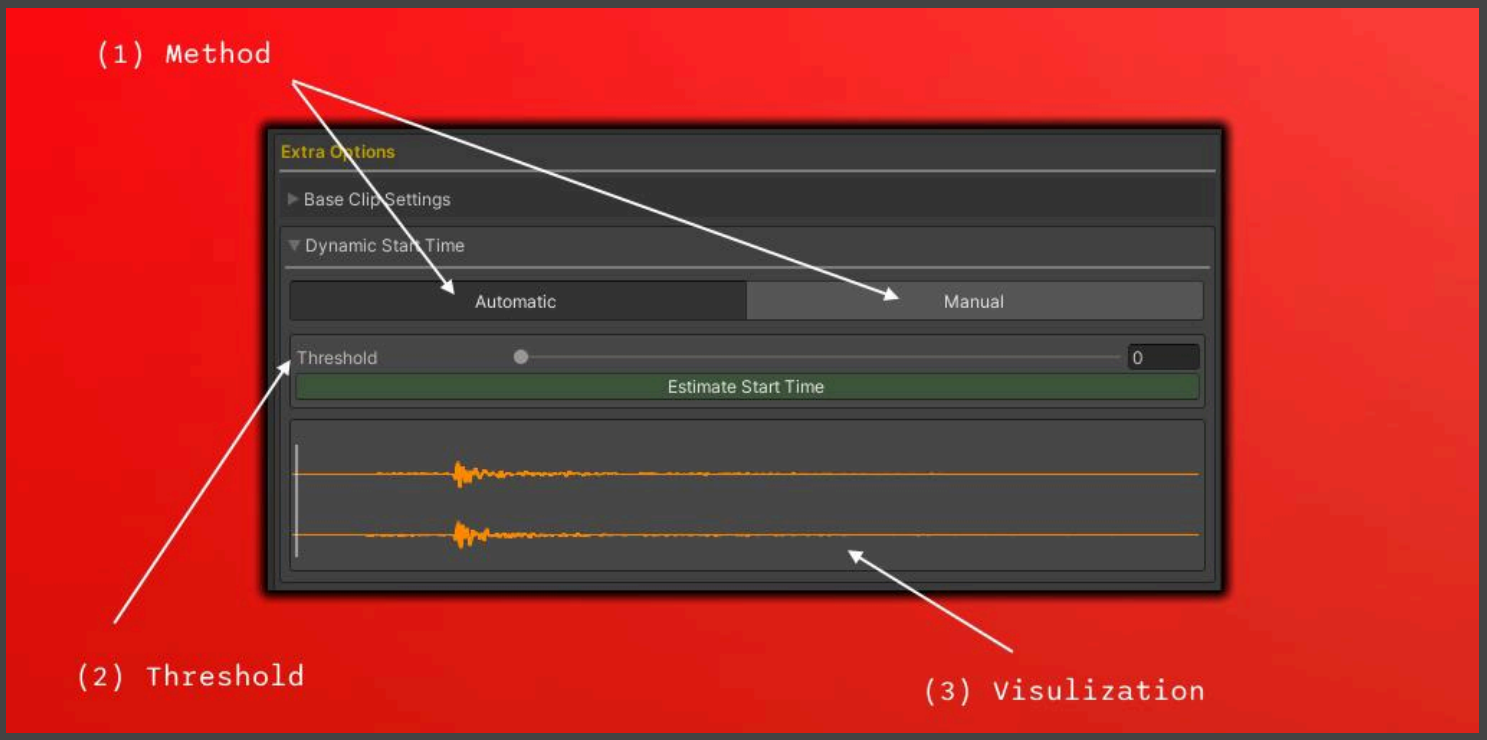
The base settings let you adjust the volume & pitch of a clip before any additional edits are made by the asset on API calls. This should be used to fine tune clips where their volumes/pitches don't quite match other clips in your project without needing to edit them outside of the project.

7.2. Dynamic Start Time

Dynamic start time is a more advanced feature that guesses when a clip actually starts from its waveform. This process isn't perfect, so you can adjust it manually as well or turn it off when calling for the clip to play through the API or in the settings for the asset.

The GUI varies based on the mode you use

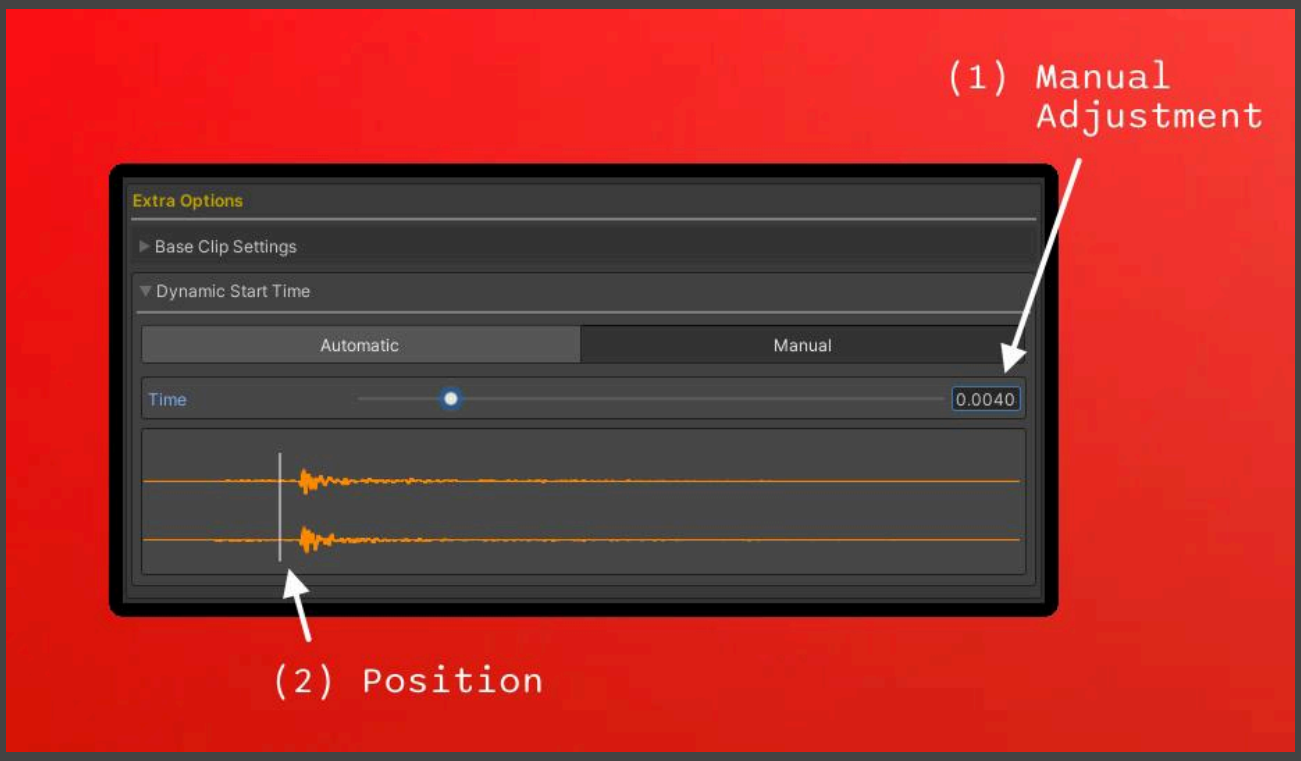
Automatic:



1. Use to toggle between automatic and manual assignment.

2. Controls how harsh the waveform check is. A higher number will ignore more of the wave when guessing the start time. On loud clips this is useful to adjust. On quieter clips you may need to adjust manually.

3. Gives a visual of where the start time is on the waveform of the audio clip. The white bar is the set start time.

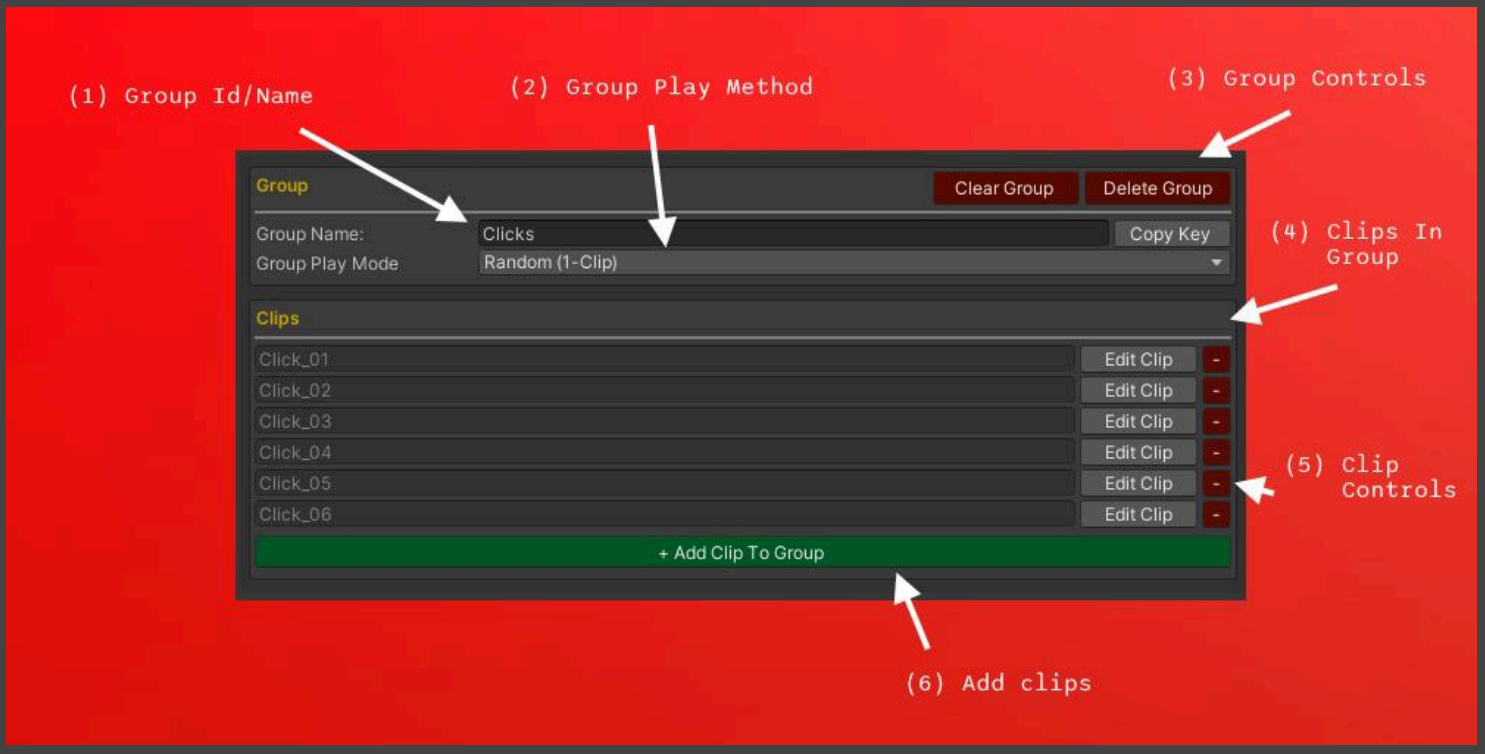


In the manual mode, you just have a slider value to adjust the start time. This will allow you to set the position shown in (2) to be anywhere on the waveform of the clip.

This setup should be used when the automatic doesn't generate the desired result or when you need some finer adjustment.

Groups

The groups tab lets you create and manage collections of clips from your library. This can be used for a variety of setups. The main reason this was made is to play a random clip from a collection of variants in one call. But it can have other applications as well.



A brief rundown of the GUI:

1. The name of the group, used in the constants class setup as well as what it shows as in the menu buttons in the left GUI element.

2. The play method for the group. The default is a random clip from the group. But you can also have it play all clips in the order of the group or combine all the clips to play at the same time.

3. The controls to clear the data in the group or to delete the group from the library all together.

4. All the clips in the group.

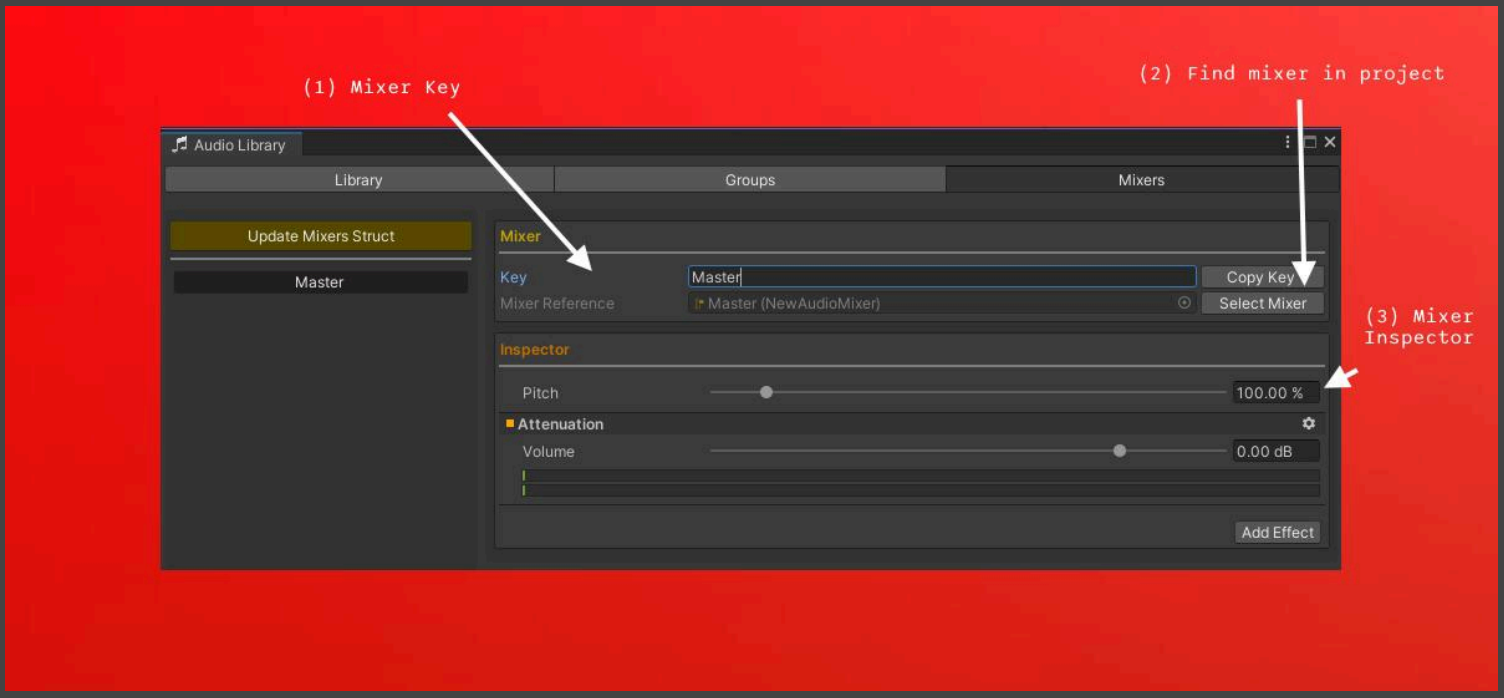
5. Controls to edit a clip entry or remove an entry entirely.

Note: you can only have 1 instance of each clip in a group.

6. Opens a search window to select a clip to add to a group. You'll only see valid options in this window.

Mixers

The mixers tab is rather bare bones, with the intent being to be able to reference them through static API when needed. You can assign keys to mixers and edit their settings if needed.



A brief rundown of the GUI:

1. The key the mixer is referred to and stored as in the constants class.
2. Selects the mixer object in the project tab.

3. Shows the normal inspector for the mixer group.

Audio Scanning

All audio in the project is automatically managed into the library for you without you needing to do anything.

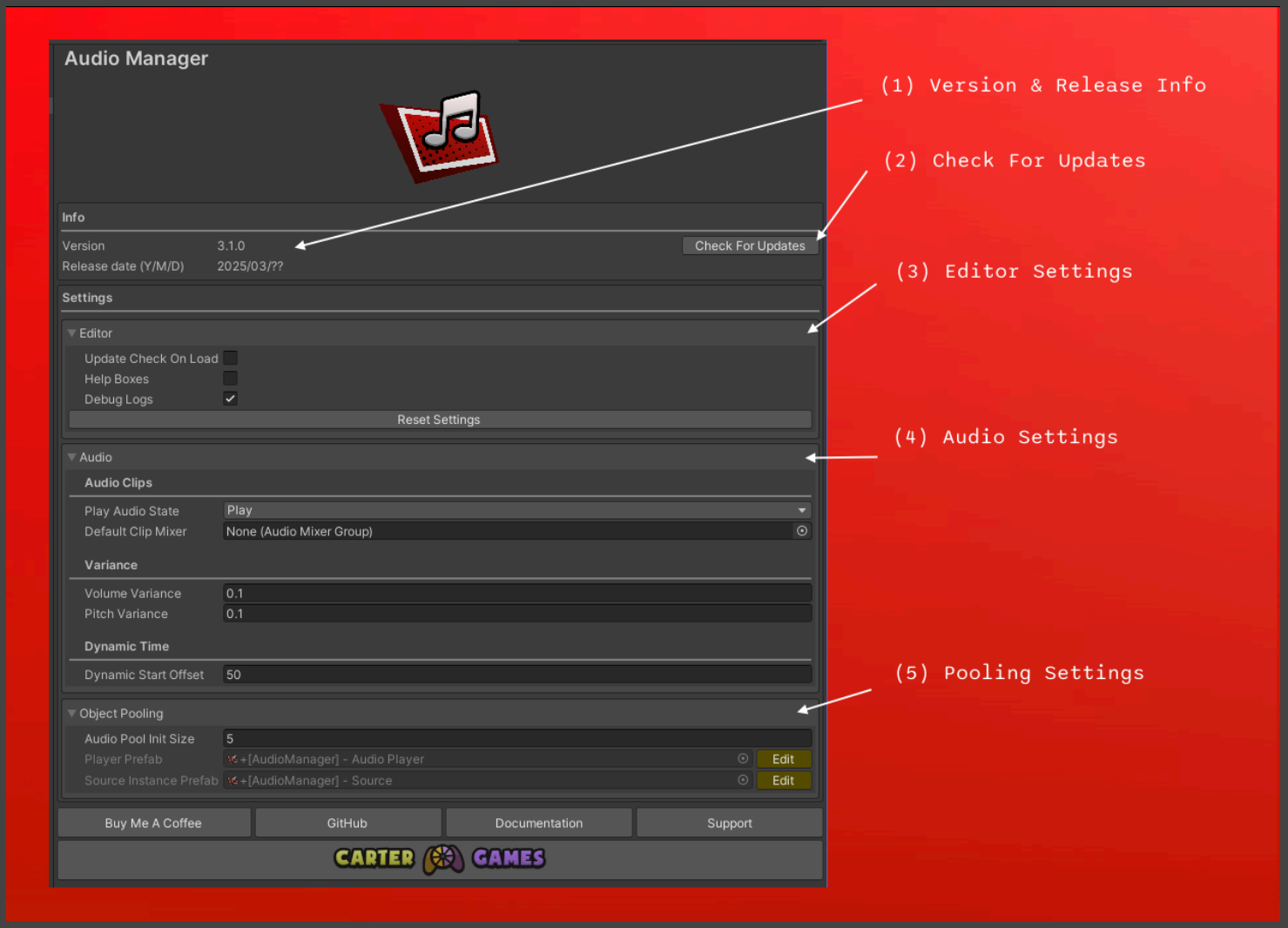
Should there be an issue where audio is missed you can manually update the audio library from the following menu item:

Tools > Carter Games > Audio Manager > Perform Manual Scan

Asset Settings

All settings for the asset can be found in the settings provider for the asset. This can be found under:

Project Settings > Carter Games > Audio Manager



A brief rundown of the settings:

1. Shows the version of the asset you are using and the release date of it. You may be asked to provide this when asking for dev support for the asset.

2. Use to check if the version of the asset you are using is the latest. Sometimes the version you are on will be behind the latest. This is more commonly the case when using the asset from the Unity Asset Store as the review process delays the releasing of updates compared to other platforms such as GitHub & Itch.io

3. Editor Specific Settings.

   These have no effect on the asset at runtime or its functionality.

   Logs from the asset that are intentional can be totally disabled in the editor to clean up the console if needed.

4. Audio Specific Settings

Setting	**Description**
Audio Play State	The default setting for audio. 99% of the time you'll want to leave this on **Play**.
Default Clip Mixer	Defines the mixer to apply to all clips in the absence of any edits with the edit modules.
Use Global Variance	Defines if the manager auto-applies the global variance settings to all clips. Global variance edit overrides this setting.
Volume Variance	Defines the global variance to apply to clip volumes when playing unless edited with an edit module
Pitch Variance	Defines the global variance to apply to clip pitches when playing unless edited with an edit module
Dynamic Start	The default auto-detect offset for the

Time offset	dynamic start time setup. Adjust if the setup is too weak at getting the right place.

5. Pooling Specific Settings

The audio manager has its own object pooling setup which is on by default.

Setting	**Description**
Audio Pool Init Size	The size of the object pool when initialized. It auto-expands when needed to accommodate your needs.
Player Prefab	The default player prefab used to play audio.
Source Prefab	The default audio source instance prefab to spawn and play audio from.

Menu Items

The asset has several menu items under the **Tools** navbar option. All of these options are under:

Tools > Carter Games > Audio Manager

Here's what they are and what they do:

Edit Settings
- Opens the settings for the asset in the project settings window.

Audio Library
- Opens the audio library editor window or focuses an existing if it is already open.

Update Asset Index
- Updates the index for the asset's scriptable objects.

Perform Manual Scan
- Manually scans the project for audio. Use if the automated system has failed to add a clip for any reason.

Reset Clips Helper Class
- Clears the Clip.cs contents class to its default value.

Reset Group Helper Class
- Clears the Group.cs contents class to its default value.

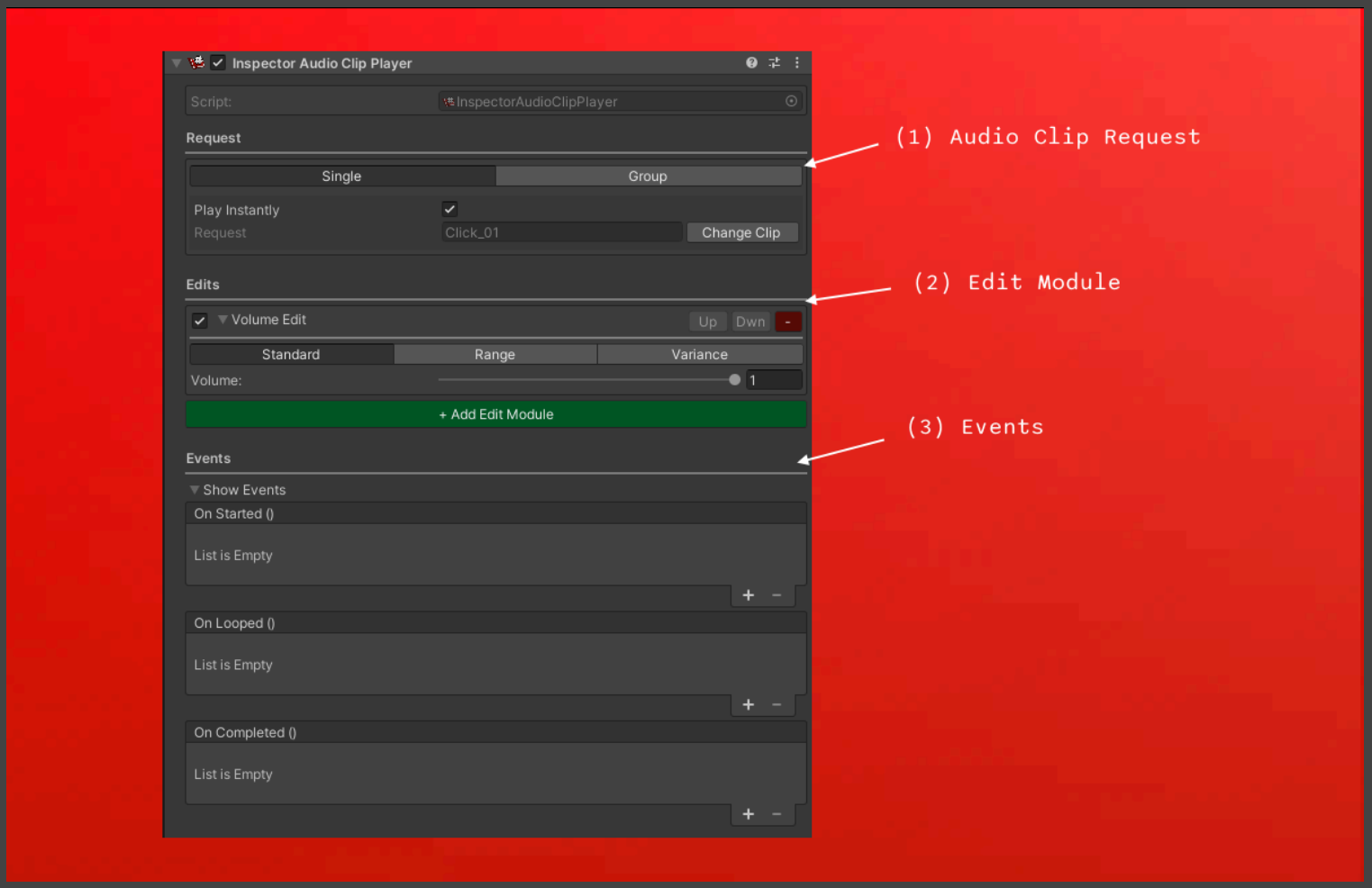
Reset Mixer Helper Class
- Clears the Mixer.cs contents class to its default value.

Reset Prefs
- Resets any asset pref settings to default values.

Pre-Built Components

The audio manager has some predefined components to handle playing audio with the manager from the inspector.

Inspector Audio Clip Player



Use to play a clip or a group of clips from the audio library.

1. Controls for the requested clip or group to play. You can have it play instantly or reference the component in a script and call it to play when you want it to.

2. You can manage the edits you want to make the player using the edit modules. Add/Remove/Toggle as many as you need.

   Note: You can only have one of each time on a single instance of the player component. Not all edit modules are available in the inspector.

3. Some Unity events that you can hook up to without the need for a script to listen to them.

Edit Modules

The edit modules setup is designed to be a modular setup to edit an audio source to the requirements of the user from a stock audio source as a base.

Whenever you call the AudioManager API, it'll use this setup behind the scenes. You can also add additional ones when calling the API or just add your own to the setup yourself for your specific use-cases.

The pre-implemented ones are:

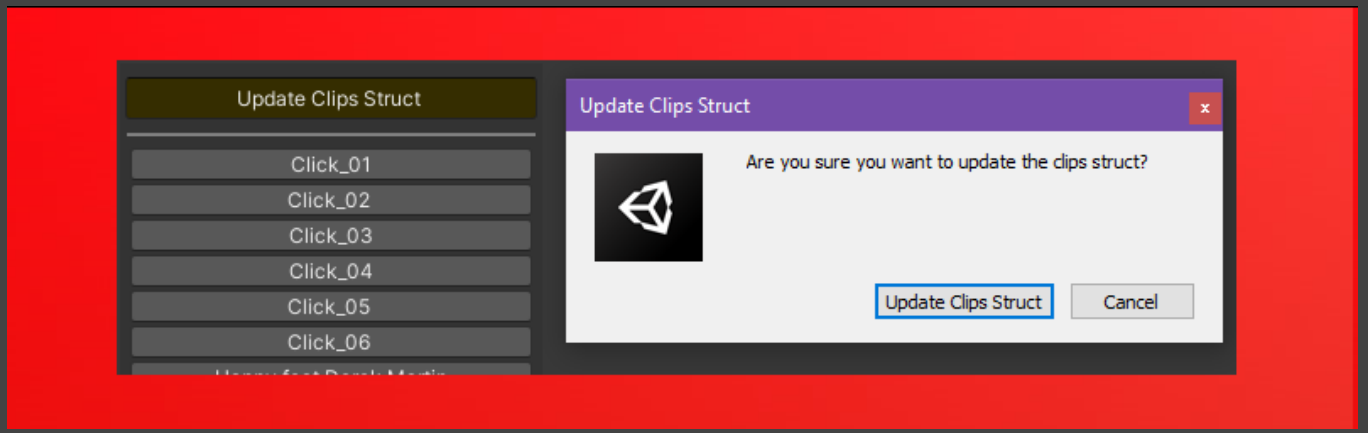
Edit Module	Description
Request	Assigns the audio clip from the library to the audio source for use.
Delay	Sets the delay before playing the audio clip when called to play.
DynamicStartTime	Sets if the audio source clip uses the dynamic start time for the requested clip. Ignores the global setting when manually set.
GlobalVariance	Sets if the audio source clip uses the global auto-variance setup. Ignores the global setting when manually set.
Loop	Assigns if the audio source should loop and how many times, if applicable.
Mixer	Assigns a mixer to the audio source. Ignores the global setting when manually set.
Mute	Sets the mute status of the audio source.
Parent	Sets the parent transform for the object the audio source is on.
Pitch	Adjusts the pitch of the audio source.
Pooling	Overrides the auto-object pooling setup so you can keep an instance

	indefinitely. Handy for music etc.
Position	Sets the transform position of the object the audio source is on.
Priority	Sets the priority of the audio source.
StartTime	Sets the time the audio source should start from. Setting this overrides any dynamic start time edits if they were to be used.
Volume	Edits the volume of the audio source component.

Only the most common setups of these have inspectors that you can use with the predefined components at the moment. I do plan to update this setup in the future to have all of them setup, but it's not a huge priority at the moment.

Constant Classes

Each setup in the audio manager such as Clip/Group/Mixer etc have their own contents class that you can generate to make selecting requests for the Audio Manager to play easier.



When updating these classes, you'll get a dialog to ensure you don't accidentally update them. The generated class looks like this once generated:

```
namespace CarterGames.Assets.AudioManager
{
   public struct Clip
   {
       public const string Click_01 =
"Click_01-f6e9b73e-ef6d-4ce4-bfed-a2d724ac590e";
       public const string Click_02 =
"Click_02-633a9bf6-13ca-44c7-bb52-6cad28268b85";
       public const string Click_03 =
"Click_03-8257234b-e650-45b2-bd24-05db88b4b4b6";
       public const string Click_04 =
"Click_04-f7867a4c-a79a-42bc-89f0-ad00540ea9bf";
       public const string Click_05 =
"Click_05-08f814d5-97ac-4d31-aed3-2b8a0026c91b";
       public const string Click_06 =
"Click_06-152e4fcb-2e4d-46a9-8944-c15366d02dee";
   }
}
```

To use this class you just call the class name followed by the entry you want. This is recommended to avoid typo's on your Audio Manager API calls.

Example:

```
private void Start()
{
   AudioManager.Play(Clip.Click_01);
}
```

Glossary Of Terms

Variance

- An offset from the value +/- by the defined amount.

  So a base value of **1** with a variance of **0.1** would result in a random number between **0.9 - 1.1**.

Scripting API

Scripting API is in a separate .pdf to save space. Please refer to it for any scripting inquiries. This is just a basic rundown to get you started:

Assemblies:

If you are using assemblies for your code base, you'll need to reference the audio manager assemblies to access the API of the asset.

Editor > CarterGames.AudioManager.Editor
Runtime > CarterGames.AudioManager.Runtime

The asset also has some shared libraries between assets. If you need to access these, you can do so from these assemblies:

Shared Editor > CarterGames.Shared.Editor.AudioManager
Shared Runtime > CarterGames.Shared.Runtime.AudioManager

Namespace:

The main namespace for the asset is
CarterGames.Assets.AudioManager

Playing a clip from the library:

```
AudioManager.Play("myClip");
```

Playing a group from the library:

```
AudioManager.PlayGroup("myGroup");
```

Error Codes

Here you can see all the error codes that the asset can send, what they mean and what to do if they occur.

#	Code String	Description	Fix Action
0	None	No error, you should never see this in any logs.	
1	Audio Disabled	Is fired if audio is called to play but the play state is set to disabled.	Change the play state to not be disabled to hide this message.
2	Clip Cannot Be Found	Is fired when the request clip cannot be found in the audio library.	Double check the clip is in the library and that there is no typo in the string entered. It is CaSe SeNsItIvE. Manually scan for audio again with a clean scan otherwise.
3	Group Cannot Be Found	Is fired when the request group cannot be found in the audio library.	Double check you have the group defined correctly and that there is no typo in the string entered. It is CaSe SeNsItIvE.
4	Mixer Cannot Be Found	Is fired when the request mixer group cannot be found in the audio library.	Double check you have the mixer defined correctly and that there is no typo in the string entered. It is CaSe SeNsItIvE.

20	Tag Cannot Be Found	Is raised when trying to get audio data from a tag that doesn't exist.	Double check you have the tag exists and is spelt correctly.
100	Prefab Not Valid	Is raised when prefab selected for a component of the audio manager isn't valid for its field.	Make sure all the components needed for the prefab are present on it.
101	Editor Only Method	Is raised if you try to use a method intended for editor only use at runtime.	Don't use the method. It will likely cause issues with the asset if you do.
102	Invalid Audio Clip Inspector Input	Is raised if an invalid input is entered into a field of an inspector audio clip player. Likely when adding values that are out of range. like -1 on a value between 0-1.	
110	Struct Generator Element Failed	Is fired if an element that was to be added to a struct generated by the asset failed to add.	See the log message for the reason why it failed and correct the issue.
111	Struct Generator No Data	Is fired if there is no data to add to a struct generated by the asset.	
112	Struct Element Name Already Exists	Is fired when a struct element variable name matches one that it has already added during the generation of the struct.	See the log message for the reason why it failed and correct the issue.

Support

Need extra help?

If you need additional help with the asset, you can contact me through email. Either directly or via the contact form on the Carter Games website.

Contact Form: https://carter.games/contact/
Email: hello@carter.games

Found a bug?

Please report any issues you find to me either via the bug report form on the Carter Games website or through an issue on GitHub. I'll try to fix these as soon as I can once reported. If I don't acknowledge your bug report, feel free to give me a nudge via email just in-case I haven't received the notification.

Bug Report Form: https://carter.games/report/
Github Issues: https://github.com/CarterGames/AudioManager
Email: hello@carter.games

Rate the asset!

If you like my asset, please do give it a rating! It's always heartwarming to know the hours I've put into this have proved helpful to someone.