Eternal - Audio Modding

Information for people who are looking to create Music, Sound Effect, and Voice Over mods.

- **General Information**
  - **Extracting Game Audio**
  - **Creating Audio Mods**
  - **Audio File Extensions**
- **Advanced Information**
  - **Audio Length**
  - **HIRC Objects**
- **DOOM 2016 Music**
  - **Entities**
  - **Events**

General Information

Information to get started.

Extracting Game Audio

It is recommended to do all of your extracting in the SAME disk drive (ex: C:\)

Extracting Tools

EternalAudioExtractor by proteh - Download
Exports WEM & OPUS audio from SND files for Windows OS.

EternalAudioExtractorLinux by PowerBall253 - Download
Exports WEM & OPUS audio from SND files for Linux OS.

Make sure you have at least 2.00 GB of storage if you plan on extracting music.
550 MB for sound effects, 50 MB for voice overs (for each language).

Extraction Instructions

Instructions are for Windows users.

- Extract the contents of **EternalAudioExtractor** somewhere on your machine.
- Launch either "**Extract_All.bat**" or "**Extract_Single.bat**" depending if you want to extract all sound files or from a single directory.
- Copy and paste the filepath of your DOOMEternal installation directory.
  (ex: **C:\Program Files (x86)\Steam\steamapps\common\DOOMEternal**)
- Input the name of the **.snd** file you want to extract (if applicable).
- Input the name of the output directory. Make sure the directory is either empty or nonexistent.
- If you are extracting music files, press **[Y]** to enable automatic .ogg conversion.
- If necessary, press **[Y]** again to enable the extraction of unused sound files.
- If everything looks correct, press **[Y]** to extract its contents.
- Press **[Y]** to close the batch script open the output folder in a new window.

Music files must be in the OGG sound format before they can be edited.

Unused game audio (represented only by ID) cannot be referenced in-game.

List of Resources

Resource Name	Information
music.snd	Contains WEM music files for Base Game & DLC1
music_patch_1.snd	Contains WEM music files for DLC2
music_patch_2.snd	Contains WEM music files for Horde Mode
sfx.snd	Contains OPUS sound effects files for Base Game
sfx_patch_1.snd	Contains OPUS sound effects files for DLC1
sfx_patch_2.snd	Contains OPUS sound effects files for DLC2
sfx_patch_3.snd	Contains OPUS sound effects files for post-DLC2
vo_English(US).snd	Contains OPUS voice over files for Base Game & DLC1
vo_English(US)_patch_1.snd	Contains OPUS voice over files for DLC2 and after
vo_French(France).snd	Contains OPUS voice over files for Base Game & DLC1
vo_French(France)_patch_1.snd	Contains OPUS voice over files for DLC2 and after
vo_German.snd	Contains OPUS voice over files for Base Game & DLC1
vo_German_patch_1.snd	Contains OPUS voice over files for DLC2 and after
vo_Italian.snd	Contains OPUS voice over files for Base Game & DLC1
vo_Italian_patch_1.snd	Contains OPUS voice over files for DLC2 and after
vo_Japanese.snd	Contains OPUS voice over files for Base Game & DLC1
vo_Japanese_patch_1.snd	Contains OPUS voice over files for DLC2 and after

vo_Polish.snd	Contains OPUS voice over files for Base Game & DLC1
vo_Polish_patch_1.snd	Contains OPUS voice over files for DLC2 and after
vo_Portuguese(Brazil).snd	Contains OPUS voice over files for Base Game & DLC1
vo_Portuguese(Brazil)_patch_1.snd	Contains OPUS voice over files for DLC2 and after
vo_Russian.snd	Contains OPUS voice over files for Base Game & DLC1
vo_Russian_patch_1.snd	Contains OPUS voice over files for DLC2 and after
vo_Spanish(Mexico).snd	Contains OPUS voice over files for Base Game & DLC1
vo_Spanish(Mexico)_patch_1.snd	Contains OPUS voice over files for DLC2 and after
vo_Spanish(Spain).snd	Contains OPUS voice over files for Base Game & DLC1
vo_Spanish(Spain)_patch_1.snd	Contains OPUS voice over files for DLC2 and after

General Information

Creating Audio Mods

See **Extracting Game Audio** before proceeding.

Instructions are for Windows users.

Audio Editing

Download an audio-editing tool of your choice, such as Audacity - **Download**

1. Import the OGG or OPUS files and edit them in your audio-editing tool.
2. Make sure that your edits match the start and end length of the audio track.
3. **SFX & VO:** Convert all of the audio you want to export to OPUS.
   (Export Audio -> OggOpus - 64 kbps for high quality)
4. **MUSIC:** Convert all of the audio you want to export to WAV. (32 bit for high quality)

FFmpeg (SFX & VO) - Eternal Mod Injector

You need the FFmpeg extension for Audacity to import/export OPUS audio files - **Download**

1. Install **ffmpeg-win-#.#.#.exe** -> Follow the installation instructions
2. Open Audacity -> Edit -> Preferences -> Libraries
3. FFmpeg Library: Locate... -> Browse... -> Find **avformat-55.dll** -> OK

Be sure to export the OPUS file in either mono or stereo depending on the format of the original file.

Sometimes, Audacity will automatically find avformat-55.dll

If the **exe** version does not work, download the **zip option** and extract it into a new folder called **ffmpeg-win-#.#.#** (depending on the version number) and extract all its contents in there. Then attempt to locate FFmpeg.

Wwise Launcher (MUSIC) - Eternal Mod Injector & idStudio

Create an audiokinetic account (use free trial) and install Audiokinetic Launcher - **Download**

Install Wwise Version **2022.1.18.8567** for the best experience.

1. Go to **Wwise Launcher** -> WWISE -> **Launch Wwise** (64-bit) -> **New** (Project Launcher) -> Provide it a name -> Select **None** (Import assets to project) -> **OK**
2. **Close** (License Manager) -> **Property Editor** (SHIFT + K) -> **Source Settings** -> Default Conversion Settings ... -> Factory Conversion Settings / Vorbis / **Vorbis Quality High** -> **OK** -> **OK**
3. **Audio File Importer** (SHIFT + I) -> Add Files... -> Select your WAV files -> **Open** -> **Import**
   (Replace any if needed)
4. **Project** -> Convert All Audio Files... -> Check **Windows** (If it is not already checked) -> **OK**
5. **File Manager** (SHIFT + F1) -> Right Click **Project Folder Directory** -> Open **Containing Folder** -> .cache\Windows\SFX (It is recommended to bookmark this directory)
6. Be sure that the exported **WEM** files end with **_3F75BDB9**

Exported **WEM** files for idStudio SFX end with **_9D0E2A85**

For idStudio mods, SFX & VO need to be converted with Wwise with the **PCM Auto Detect High** Default Conversion Setting. See: **https://wiki.eternalmods.com/books/eternal-idstudio/page/how-do-i-create-idstudio-mods**

Injecting Audio

1. In your DOOMEternal\Mods folder, create a folder called named based on where the file was originally found.
   (**ex: music_patch_1**)
2. Copy your converted WEM file(s) in there and make sure they have the same name as extracted files.
   (**ex: samuelsbase_music_main_heavy_0_0_id#15098487.wem**)

3. Run the **EternalModInjector** like normal.

Audio File Extensions

These are the audio file types you will be interacting with.

.ogg

Extracted music files from "**music**", "**music_patch_1**", & **"music_patch_2"** will be converted to this format if they are to be edited.
OGG files **CANNOT** be re-injected into DOOM Eternal.
Example:
`cultist_base_music_main_heavy_0_id#156134790.ogg`

.opus

Extracted sound effect & voice over files from "**sfx**", "**sfx_patch_1**", "**vo_English(US)**", etc will be in this format.
OPUS files **CAN** be re-injected into DOOM Eternal.
Example:
`ammo_bfg_id#373311939.opus`

.wav

Edited music files must be converted into the WAV format before they can be converted into WEM files.
WAV files **CANNOT** be re-injected into DOOM Eternal as music files, but they **CAN** as sound effects & voice overs.
Example:
`cultist_base_music_main_heavy_0_id#156134790.wav`

.wem

After the edited music files have been converted into the WEM format, they **CAN** be re-injected into DOOM Eternal.
Example:
`cultist_base_music_main_heavy_0_id#156134790.wem`

.snd

These contain the audio files. They **CANNOT** be directly modified by a conventional audio editor. Instead, tools are used to extract individual audio audio files.
Example:
`music.snd`

.pck

These contain the audio resources. They **CANNOT** be directly modified by a conventional audio editor.

However, **Fusion Tools** can directly modify PCK files.
Example:
`mus.pck`

> If a **WEM** file has been exported correctly (Vorbis Quality High), they should have "**_3F75BDB9**" before its WEM extension. You will need to delete the "**_3F75BDB9**" in the WEM file's name before you can inject it into DOOM Eternal.

> **PCK** files modified by **Fusion Tools** are not supported by **DEternal_loadMods**, which is a program that allows certain mods type to be injected.

> See **Creating Audio Mods** for more information.

Advanced Information

Advanced information that covers Fusion Tools and Wwise.

Advanced Information

Audio Length

> See **Extracting Game Audio** & **Creating Audio Mods** before proceeding.

> This is an **Advanced Guide**, you are expected to have a decent understanding of how audio (primarily music) flows in DOOM Eternal.

Backup your PCK files

Find the location of whichever **PCK** file you want to edit - **mus**, **sfx**, **vo_english**, etc.
(ex: **C:\Program Files (x86)\Steam\steamapps\common\DOOMEternal\base\sound\soundbanks\pc**)

Copy them over to a safe location, this will be the file that you will edit.
You may copy an additional **PCK** file if you want to backup the original.

Fusion Tools

Fusion Tools is an advanced sound editor. For the case of DOOM Eternal, you will use it to export/import the **BNK** that allows you to edit music length and their start/stop positions.

Fusion Tools - Download

Exporting the BNK file (Into the XML Format)

Extract the contents of Fusion Tools and put **FusionTools.exe** somewhere safe.

- Launch **FusionTools.exe**
- **Wwise Editor**
- **File options** (top left of the window)
- **Open File**
- Navigate to where you copied the **PCK** file that you want to edit
  (You may get a warning that the **PCK** file does not have any WEMs, just select okay and ignore it)

- **Show BNKs embedded in the file** (3rd option on the top bar)
- In the new window, right click any option available -> **Edit BNK** (**mus.pck** only has 1 Embedded BNK)
- In the new Window, select: **Export/Import HIRC section to/from an XML file** (4th option on the top bar)
- **Export HIRC** (Export it somewhere safe, do not change the name of the file)

Editing Sound Properties

Proceed after your sound files are in the proper, inject-able format.

Save a copy of the XML file if you want to reference the unmodded piece.

Open the XML file in any XML viewer. **Notepad++** works just fine - **Download**

- Find the sound IDs that appear at the end of each extracted sound file.
  For music, you will see 2 of them, they are the **SourceID**.
- The parent ID, or just **ID**, encompasses the entire track.
- There are 2 locations where the **SourceDuration** is held for each track. The duration is measured in milliseconds.
- Each can be changed to the desired amount, but for simplicity's sake, they should be identical.

```
<object type="MusicTrack">
    <field type="byte" name="HircType" value="11" />
    <field type="int32" name="Size" value="150" />
    <field type="uint32" name="ID" value="504369478" />
    <field type="byte" name="MusicFlags" value="0" />
    <list type="TrackSource" name="TrackSources" count="1">
        <object type="TrackSource">
            <field type="uint32" name="PluginID" value="262145" />
            <field type="byte" name="StreamType" value="2" />
            <field type="uint32" name="SourceID" value="280074348" />
            <field type="int32" name="InMemoryMediaSize" value="4266" />
            <field type="byte" name="SourceBits" value="0" />
        </object>
    </list>
    <list type="TrackItem" name="TrackPlaylist" count="1">
        <object type="TrackItem">
            <field type="uint32" name="ID" value="0" />
            <field type="uint32" name="SourceID" value="280074348" />
            <field type="uint32" name="EventID" value="0" />
            <field type="double" name="PlayAt" value="-1800" />
            <field type="double" name="BeginTrimOffset" value="1800" />
            <field type="double" name="EndTrimOffset" value="0" />
            <field type="double" name="SourceDuration" value="83999.97916666666" />
        </object>
    </list>
```

```
    <list type="uint32" name="ChildIDs" count="1">
        <field type="uint32" name="ID" value="504369478" />
    </list>
    <field type="double" name="AkMeterGridPeriod" value="1000" />
    <field type="double" name="AkMeterGridOffset" value="0" />
    <field type="float" name="AkMeterTempo" value="120" />
    <field type="byte" name="AkMeterTimeSigNumBeatsBar" value="4" />
    <field type="byte" name="AkMeterTimeSigBeatValue" value="4" />
    <field type="byte" name="AkMeterInfoFlag" value="0" />
    <list type="byteArray" name="Stingers" count="0" />
    <field type="double" name="Duration" value="82199.9791666667" />
    <list type="MusicMarker" name="MusicMarkers" count="2">
        <object type="MusicMarker">
            <field type="uint32" name="ID" value="43573010" />
            <field type="double" name="Position" value="600" />
            <field type="string" name="Name" value="" />
        </object>
        <object type="MusicMarker">
            <field type="uint32" name="ID" value="1539036744" />
            <field type="double" name="Position" value="77400" />
            <field type="string" name="Name" value="" />
        </object>
    </list>
</object>
```

- **PlayAt** determines when the track actually starts. For the most part, this is unnecessary, so it should be changed to 0. It is measured in milliseconds.

- **BeginTrimOffset** & **EndTrimOffset** determine when the track plays and when it cuts off. The negative amounts means it triggers from the end of the duration to the specified location. It is recommended to set both of those values to 0. They are measured in milliseconds.

- **MusicMarkers** are what determines when the current track starts and when the next track will start.
  Change their positions to the appropriate amounts. They are measured in milliseconds.

ID **43573010** is when the current track starts
ID **1539036744** is when the next track starts (the current track ends)

Keep in mind that **MusicMarkers** do not cut off the music. The **BeginTrimOffset** & **EndTrimOffset** do that.

- **AutomationItems** cause the music to fade in or fade out.
  For most instances, they are unnecessary and can be removed. They are measured in seconds.

```
<field type="uint32" name="NumSubtrack" value="1" />
<list type="AutomationItem" name="AutomationItems" count="1">
    <object type="AutomationItem">
        <field type="uint32" name="ClipID" value="0" />
        <field type="uint32" name="AutoType" value="3" />
        <list type="GraphPoint" name="GraphPoints" count="2">
            <object type="GraphPoint">
                <field type="float" name="From" value="0" />
                <field type="float" name="To" value="0" />
                <field type="uint32" name="Interpolator" value="6" />
            </object>
            <object type="GraphPoint">
                <field type="float" name="From" value="0.37365597" />
                <field type="float" name="To" value="1" />
                <field type="uint32" name="Interpolator" value="9" />
            </object>
        </list>
    </object>
</list>
```

Change the contents from the image above to the one below.

```
<field type="uint32" name="NumSubtrack" value="1" />
<list type="AutomationItem" name="AutomationItems" count="0" />
```

It might be helpful to keep**AutomationItems** and alter them based on the modded track.

To modify music volume, go to the **MusicSequence** object that contains the **DirectParentID** for each **MusicTrack** and find "Prop".

```
<list type="Prop" name="Properties" count="1">
    <object type="Prop">
        <field type="byte" name="ID" value="0" />
        <field type="uint32" name="Value" value="1082130432" />
    </object>
</list>
```

Change the field type to **"float"**, from **"uint32"** and the value to an existing value of a different number. The higher the value, the louder the track, and vice versa.

```
<list type="Prop" name="Properties" count="1">
    <object type="Prop">
        <field type="byte" name="ID" value="0" />
        <field type="float" name="Value" value="1086324736" />
    </object>
</list>
```

The following are valid float values for Prop (from quietest to loudest):

1056964608
1065353216
1073741824
1077936128
1080033280
1082130432
1083179008
1084227584
1086324736
1090519040
1093664768
1094713344
1098907648
1099431936

If there are no Props or it is set to 0, you can add it. Here is a sample you can copy and paste into your XML file.

```
<list type="Prop" name="Properties" count="1">
    <object type="Prop">
        <field type="byte" name="ID" value="0" />
        <field type="float" name="Value" value="1086324736" />
    </object>
</list>
```

Music IDs are organized by category (Ambient, Light, Heavy, etc).
They are measured alphabetically, in the same format as File Explorer if organized by name.

It is up to your best judgment to listen by ear if you want to match the length & music markers accurately.

Importing the XML file into PCK

- After changes are made to the exported XML file, save it, then go back to the last Fusion Tools window.
- **Export/Import HIRC section to/from an XML file** (4th option on the top bar)
- **Import HIRC**
- Go to the first **Wwise Editor - Ready** window -> **File options**
- **Save file** -> Save as the same name as the PCK file that was edited
  (ex: **mus.pck**)

As of current time, the **Eternal Mod Injector** does not support **PCK** files.
The modded **PCK** file must be manually switched between the vanilla file.

Alternatively, the **Eternal PCK Installer** script can be used for easier .pck file replacement/restoration.

See Also

- **HIRC Objects (Advanced Guide)**

HIRC Objects

See **Audio Length (Advanced Guide)** before proceeding.

HIRC (Hierarchy) Objects

These are the components to the PCK files. They are separated by **<object>** in the exported XML document.

Music

Attenuation

Determines the force reduction of the overall audio. It is measured by "Curves" which can be scaled by "GraphPoints" using an "Interpolator".

MusicTrack

Actual components to each individual music track. It determines the "Duration" and "TrimOffSets" of the track.

MusicSegment

Immediately following a "MusicTrack" is its "MusicSegment". It also determines the "Duration", but it establishes "Music Markers" which actuates when the track starts and ends.

MusicSequence

Contains the list of all "MusicTracks" that will play in the sequence. It determines the the "TransitionRules" and "Destination Rules" for each Music Track.

MusicSwitchContainer

Decides when the "MusicTracks" start playing. It is structured much like a "MusicSequence".

EventAction

Directly responsible for starting and stopping the music.

Event

Immediately following an "EventAction". This is the individual event that holds the ID which will start or stop.

DOOM 2016 Music

Music from DOOM 2016 can be played in DOOM Eternal without changing any of the music files.

Entities

The idSoundEntity must be activated first. Then one of the idMusicEntity entities can be activated.

The **setMusicState** eventCall will not work, they must be triggered either by an **idTarget_Count** entity or **activateTarget** eventCall.

The **idMusicEntity** playing before activating the DOOM 2016 idSoundEntity **might overlap,** even persisting between levels. Deactivate it first by switching it to **music_ghost_states/none**.

```
entity {
    entityDef sound_sound_musicentity_2016 {
    inherit = "sound/soundentity";
    class = "idSoundEntity";
    expandInheritance = false;
    poolCount = 0;
    poolGranularity = 2;
    networkReplicated = false;
    disableAIPooling = false;
    edit = {
        temporarySoundEvent = true;
        soundOcclusionBypass = true;
        startEvents = {
            num = 1;
            item[0] = "Play_music_doom_2016_04_surface_1";
        }
        stopEvents = {
            num = 1;
            item[0] = "Play_music_doom_2016_04_surface_1";
        }
    }
}
}
```

```
entity {
    entityDef sound_sound_musicentity_2016_heavy {
    inherit = "sound/musicentity";
    class = "idMusicEntity";
    expandInheritance = false;
    poolCount = 0;
    poolGranularity = 2;
    networkReplicated = false;
    disableAIPooling = false;
    edit = {
        spawnPosition = {
            x = 1;
            y = 1;
            z = 1;
        }
        initialState = "music_doom_2016_combat_state/heavy";
    }
}
}
```

```
entity {
    entityDef sound_sound_musicentity_2016_light {
    inherit = "sound/musicentity";
    class = "idMusicEntity";
    expandInheritance = false;
    poolCount = 0;
    poolGranularity = 2;
    networkReplicated = false;
    disableAIPooling = false;
    edit = {
        spawnPosition = {
            x = 1;
            y = 1;
            z = 1;
        }
        initialState = "music_doom_2016_combat_state/light";
    }
}
}
```

```
entity {
    entityDef sound_sound_musicentity_2016_ambient {
    inherit = "sound/musicentity";
    class = "idMusicEntity";
    expandInheritance = false;
    poolCount = 0;
    poolGranularity = 2;
    networkReplicated = false;
    disableAIPooling = false;
    edit = {
        spawnPosition = {
            x = 1;
            y = 1;
            z = 1;
        }
        initialState = "music_doom_2016_combat_state/map_start";
    }
}
}
```

```
entity {
    entityDef sound_sound_musicentity_2016_end {
    inherit = "sound/musicentity";
    class = "idMusicEntity";
    expandInheritance = false;
    poolCount = 0;
    poolGranularity = 2;
    networkReplicated = false;
    disableAIPooling = false;
    edit = {
        spawnPosition = {
            x = 1;
            y = 1;
            z = 1;
        }
        initialState = "music_doom_2016_combat_state/end";
    }
}
}
```

Events

Boss & Cutscene music are not available.

Start & Stop Events

```
play_music_doom_2016_01_intro
play_music_doom_2016_02_resops
play_music_doom_2016_03_foundry
play_music_doom_2016_04_surface_1
play_music_doom_2016_05_argent
play_music_doom_2016_06_bka
play_music_doom_2016_07_surface_2
play_music_doom_2016_08_bfg
play_music_doom_2016_09_lazarus
play_music_doom_2016_10_bkb
play_music_doom_2016_11_bkc
play_music_doom_2016_12_polar_core
play_music_doom_2016_13_titan
play_music_main_menu_01
```

Music States

Only **map_start**, **light**, **heavy** and **end** will work for every Start & Stop Events

```
music_doom_2016_combat_state/end
music_doom_2016_combat_state/heavy
music_doom_2016_combat_state/light
music_doom_2016_combat_state/light_end
music_doom_2016_combat_state/map_start
music_doom_2016_combat_state/none
music_doom_2016_combat_state/reveal
music_doom_2016_combat_state/rune_challenge_end
music_doom_2016_combat_state/rune_challenge_start
music_doom_2016_combat_state/vulnerable
```