.scd Editing for Music/SFX: A (Mostly) Comprehensive Guide

(written by: gator.exe [taco])

[updated: August 11, 2026 - see changelog]

[Table of contents]

I've tried to format, adjust, and streamline the information in this document as much as I've been able, and give the more frequented parts the ability to stand on their own. As such, there may be a lot of repetition between sections. I apologise if anything feels odd, lacking, or isn't helpful.

Before I get into it, I do want to give massive thanks to ocealot for VFXEditor, goaaats for Quick Launcher, Fragmenterwork's FFXIV Explorer, all the driving forces behind Penumbra, ff-meli / perchbird for the Orchestrion plugin, and everyone else who's working behind the scenes. None of this would be possible without you all.

Contact info

If you have any questions, corrections, concerns, hate mail, want to share what you know, or simply wish to say thanks, reach out to **@caffeinatedgator** on Bluesky. Responses may be slow, but I promise I do check my DMs there.

You may also wish to check the Students of Baldesion server and the official VFXEditor thread in the Dalamud server for additional support or to see if something has been answered before.

If this guide helped you and you want to give some monetary support, I have a Ko-Fi page set up. I've spent many, many, *many* hours on all my guides and research documents; donations - no matter how little - go a long way. Thanks for your consideration!

Getting started

This guide uses the **VFXEditor plugin** (and, by extension, XIV Quick Launcher); make sure you have this installed.

You also will want to find a good conversion software or website to turn your custom audio file into **OGG**, **WAV**, or **HCA**, the formats that are usable by the game's native **.scd** container.

Highly recommended to install is the **Orchestrion plugin**. This helps with testing of any custom BGM.

Finally, I'll be referencing **Audacity** occasionally, but this program is mostly optional to install. If you're doing any manual edits to your sound - like custom loop points, trimming audio, or working with dynamic combat music - you'll want to have this or similar editing software on hand.

To navigate this document: in addition to the table of contents above, you can click the three-line (hamburger) icon to the left to quickly jump to a relevant section.

For those who wish to jump straight to how to work with BGM and sounds, go here.

Preparing audio with Audacity

For basic BGM, **this is an optional step** for manual and precise edits of your sound.

For dynamic combat music (e.g. PvP, ARR hard-mode dungeons, some alliances, Bozja/Eureka, and BGM unaffected by the Enable normal battle music. setting in FFXIV's *System Configuration* window), you will need to do these edits before moving on. Click here to move to the corresponding section.

If you wish to know about samples and where to find them - seen in programs like FFXIV Data Explorer or Sebastina's Voice Pack Creator - go here.

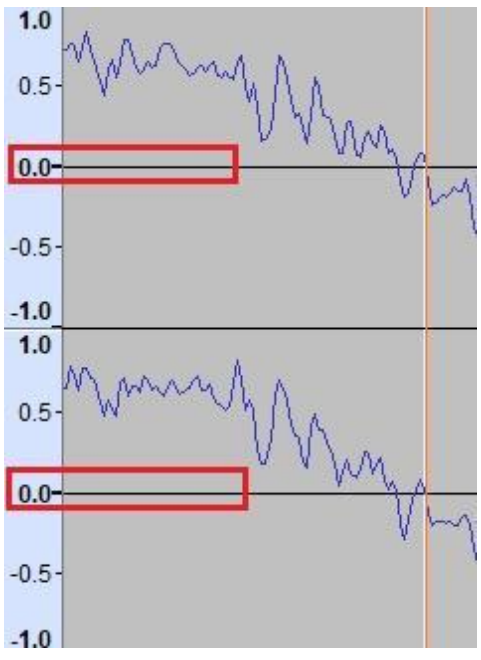
Standard BGM and sounds

Load your chosen file into the audio editor with ***File > Open...***

Edit your file as necessary with cuts, fades, Paulstretches, or what have you. Please refer to Audacity's own manual if you're unsure of any specifics beyond what's mentioned here.

If you're doing cuts and you plan to loop at that cut point, do your best to trim where all available channels will have zero crossing, or near it, so you don't get pops in your audio. Zero crossing is where the wavelengths of the audio cross the 0.0 line; you may have to zoom in quite a bit to see this line.

In the zoomed-in example image below, the 0.0 line is boxed in **red**, and where both (stereo) channels cross it at the same time is the vertical **orange**:



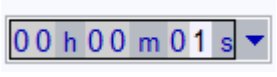
For custom loops, I'd also suggest finding where your loop point would be the least conspicuous, while also preserving the original introduction in the song, which usually slightly differs from its repeated counterpart later on. It's recommended to mark your loop points with labels - ***Edit > Labels > Add Label at Selection*** once you've clicked at the spot you want it. Labels will help to avoid having to search for these spots again. You can type something to give each label a name, if you want.

Important: If you're working with BGM that you don't want to loop, try your best to match the sample length of your file to the one you're changing, whether that be cutting out extraneous music or adding in silence as a buffer. This is because FFXIV sets the repeat timing specific to that non-looped BGM, and going over or under will cause issues in those forced repeats of your custom BGM.

For the quick and dirty of how to check the sample length:

- Go into VFXEditor, click the **scd** tab at the top, and click the icon to the right the **Loaded Scd** field to search for your BGM. Click the checkmark next to the file name once you've found it.

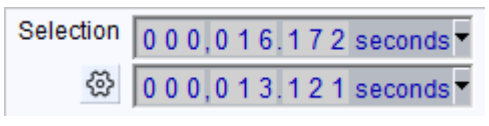
- Click *Audio 0*. Next to the built-in player that popped up to the right, there's a floppy disk button to its right. Click this button to bring up a window to save the audio, then open the audio up in Audacity.

- Click the arrow button to the right of the time field 

and click **samples**. Press the **End** button on your keyboard (or click at the end of the waveform of the audio). Now you have the target sample value to aim for.

For non-looped audio, feel free to skip ahead.

For looping audio, have your audio editor show the project in **seconds + milliseconds**. Click the arrow▾next to the time values and pick it from the dropdown menu:



We want to keep track of three values in particular - the *starting loop point*, the *length of the loop*, and then the *end loop point*. In our case, the **top** field is the **start point** and the **bottom** field is the **loop length**. We'll use them later when we're actually editing the SCD.

To get length, click your loop start point, hold Shift on your keyboard, then click where your loop end point is. This will highlight that selection and show the seconds value for it.

For the **end point**, either press **End** on your keyboard or click where your ending loop point is and read the **top** field. (Or add start and length together to get it. That works, too.)

Once you've done all you need to do in Audacity, export the final product (***File > Export Audio***) in one of the following formats:

- If you're not worried about custom loop points, then **OGG** at a quality of 6-7 at 44100 Hz is fine.

- If you want potentially more precise loops and you're not averse to text-only interfaces, export in any lossless format (**WAV** or **FLAC** recommended) and then refer to the following section on downloading and using FFmpeg.

- Or if you're just working with standard sound effects, you can export as **WAV**. The recommended encoding is currently **Signed 16-bit PCM**. Note that, since late 2024, WAV files have not been behaving consistently when used with loops. Therefore, it's recommended to use OGG for that purpose or see the following section on FFmpeg usage for the time being.

If it's **OGG**, and you have custom loop points and want to avoid the FFmpeg codec method, you might consider using the highest available quality setting to make loops a little easier to manage. This will increase your file size, so keep that in mind if you're working with large mod packs.

> The more you know: The relevance of OGG quality and sample rate has to do with something called "pages" that this format creates for each unique file. In simple terms, for our purposes, pages establish the accuracy of where VFXEditor is able to set loop times. A lower quality and sample rate will reduce this accuracy, so it's - more often than not - better to have it higher. Upsampling is a point of contention all its own, whether it actually helps or hurts, but this is an argument best left for audiophiles and those with high-end sound systems.

Please note that FFXIV has seemingly terminated support for WAV sounds in files intended for BGM. Therefore, playback in that situation will result in silent audio. It's possible that there's a native workaround or mod loaders can to find a way to sidestep this, although it's unlikely in either case.

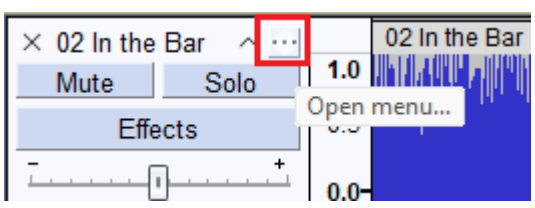
Non-standard BGM

Heads up: Replacements of this type in-game **are limited to areas that already support dynamic combat music**. This means you could replace Labyrinth of the Ancients with dynamic music, but not something like The Puppets' Bunker. This is currently a game limitation.

For our non-standard combat BGM, we'll be making it four-channel audio. Normally, this would be six channels to fit how the game does it, but import of six channels is currently broken in VFXEditor with no ETA on a solution.

Four channels often requires two separate songs, and I'll be working with that assumption for this example, so import both files you want to use for it. Use ***File > Import > Audio...*** to get them both in the same workspace; ***File > Open...*** will separate them, which we don't want.

Now we want to *split stereo channels into mono*. FFXIV has a unique setting that will mix these mono channels into stereo itself and also determine what plays in combat or not, according to what channels they're in. Your **out-of-combat music** will be the **first two channels**, while the **combat music** will be the **second two channels**. Before we do anything else, make sure the out-of-combat music is set above the combat music. Then, click the three dots **...** (or right click that area) next to the file name to the left of your waveform to bring up a drop-down menu, and click ***Split Stereo to Mono***. Do this for both of your songs. If you did it right, you'll have four separate tracks.



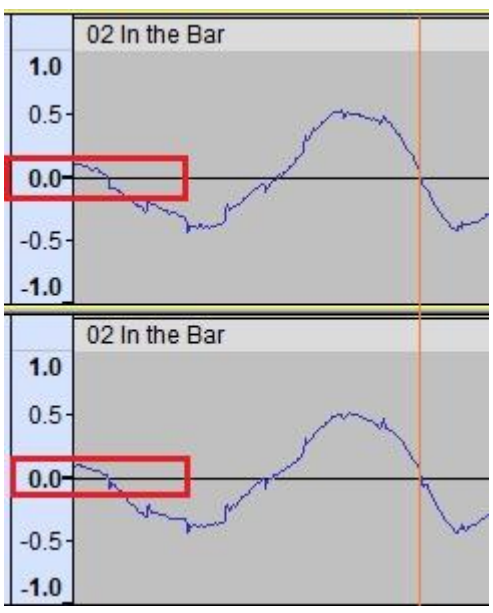
> Side note: In FFXIV native files, the fifth and sixth mono channels would be something like a beat marker, most often a cymbal crash with a bass drum. These channels would overlap the others and provide a more natural transition, rather than cutting directly between them. If you really wanted, you could do something similar and just mix it into each song so you'd get the same effect.

Edit your file as necessary with cuts, fades, Paulstretches, or what have you. Please refer to Audacity's own manual if you're unsure of any specifics beyond what's mentioned here.

Make sure things are aligned how you want them to be, so that any transitions you had in mind don't come across as too awkward. We'll set where those transitions are later, although it most likely won't be exact in the final product. Not much you can do about that without SE's proprietary software.

If you're doing cuts and you plan to loop at that cut point, do your best to trim where all available channels will have zero crossing, or near it, so you don't get pops in your audio. Zero crossing is where the wavelengths of the audio cross the 0.0 line; you may have to zoom in quite a bit to see this line.

In the example image below, the 0.0 line is boxed in **red**, and where two mono channels cross it at the same time is the vertical **orange**:

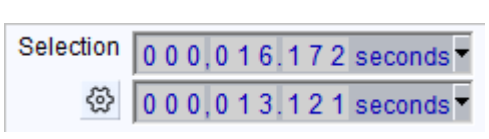


If you're planning on going the custom loop route, I'd also suggest finding where your loop point would be the least conspicuous, while also preserving the original introduction in the song, which usually slightly differs from its repeated counterpart later on. It's recommended to mark your loop points with labels - ***Edit > Labels > Add Label at Selection*** once you've clicked at the spot you want it. Labels will help to avoid having to search for these spots again. You can type something to give each label a name, if you want.

You'll also want to put labels at each place you want the music to transition according to combat status. As a guideline, FFXIV marks them **every 3-4 seconds, on average**.

Labels aren't strictly necessary, but you'll save yourself a major headache later.

Now have your audio editor show the project in **seconds + milliseconds**. Click the arrow▾next to the time values and pick it from the dropdown menu:



To start with, we want to keep track of three values in particular - the *starting loop point*, the *length of the loop*, and then the *end loop point*. In our case, the **top** field is the **start point** and the **bottom** field is the **loop length**. We'll use them later when we're actually editing the SCD.

To get length, click your loop start point, hold Shift on your keyboard, then click where your loop end point is. This will highlight that selection and show the seconds value for it.

For the **end point**, just click where your ending loop point is and read the **top** field. (Or add start and length together to get it. That works, too.)

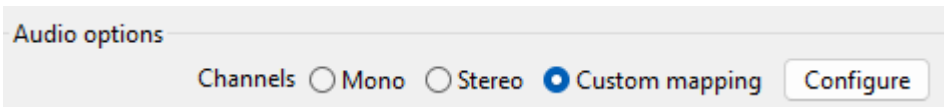
Also write down the values at each transition that you *hopefully* marked with labels. If they're not marked, well, maybe we've learned something.

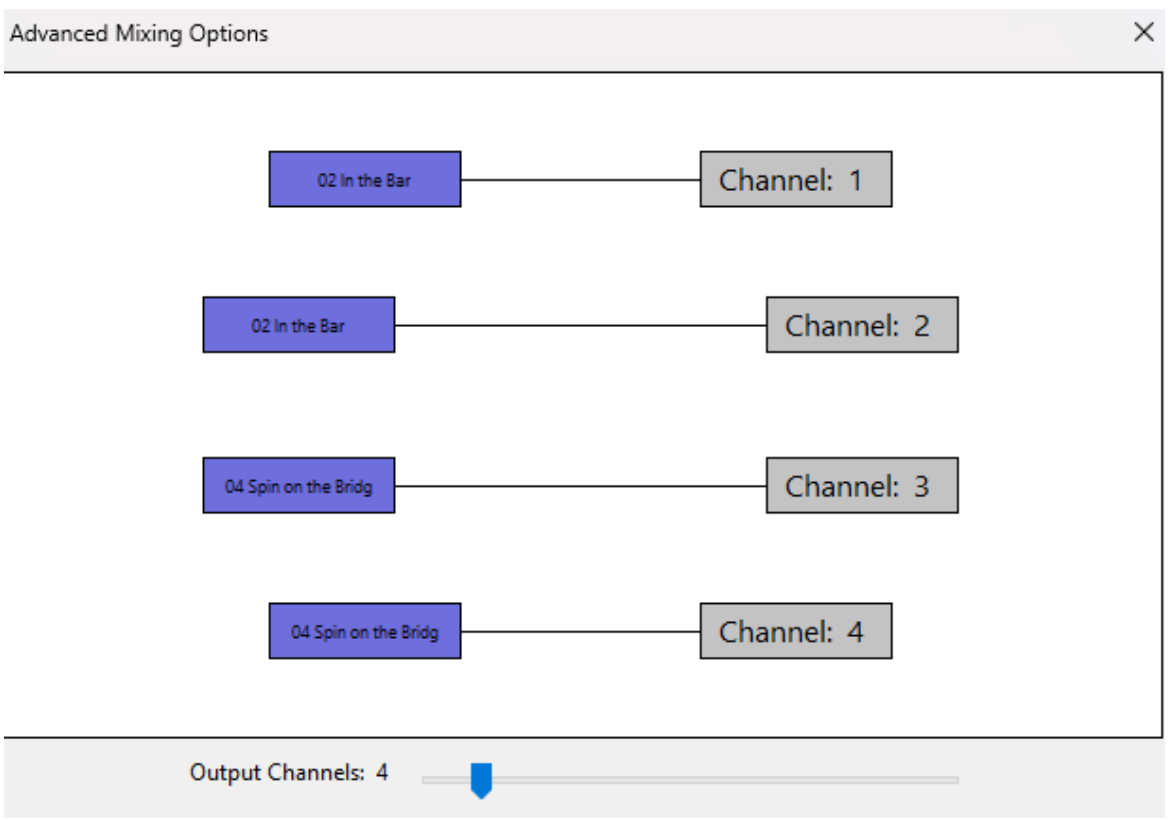
Once you've done all you need to do in Audacity, export the final product as **OGG** (***File > Export Audio***). A quality of 6-7 at 44100 Hz is fine for most cases. WAV files are no longer supported in files intended for BGM; thus, OGG is your only choice until a workaround is discovered.

If you have custom loop points and want to avoid the FFmpeg codec method, you might consider using the highest available quality setting to make loops a little easier to manage. This will increase your file size, so keep that in mind if you're working with large mod packs.

Additionally, we need to click the **Custom mapping** radio button during the export process and set the **Output Channels** to **4** when clicking **Configure**. You may need to connect the tracks to the channels if they aren't: click the track and then the channel to tie it to in order to connect them. (They'll be in order already.) Deselect them both to avoid accidentally connecting to others, and repeat for the rest of the channels.

> Side note: Because we're using more channels, that does mean the size will be **double** what you would normally have (or 2.5x or 3x, if six-channel becomes usable later). Something like a four-minute song with four channels at the highest OGG quality could end up as 30MB, depending on initial imported bitrate. You'll have to decide if that extra size is worth it or acceptable with regards to the end user, as well as how much a lower-quality export will affect loop and transition points.





> The more you know: The relevance of OGG quality and sample rate has to do with something called "pages" that this format creates for each unique file. In simple terms, for our purposes, pages establish the accuracy of where VFXEditor is able to set loop times. A lower quality and sample rate will reduce this accuracy, so it's - more often than not - better to have it higher. Upsampling is a point of contention all its own, whether it actually helps or hurts, but this is an argument best left for audiophiles and those with high-end sound systems.

By the way, what are samples?

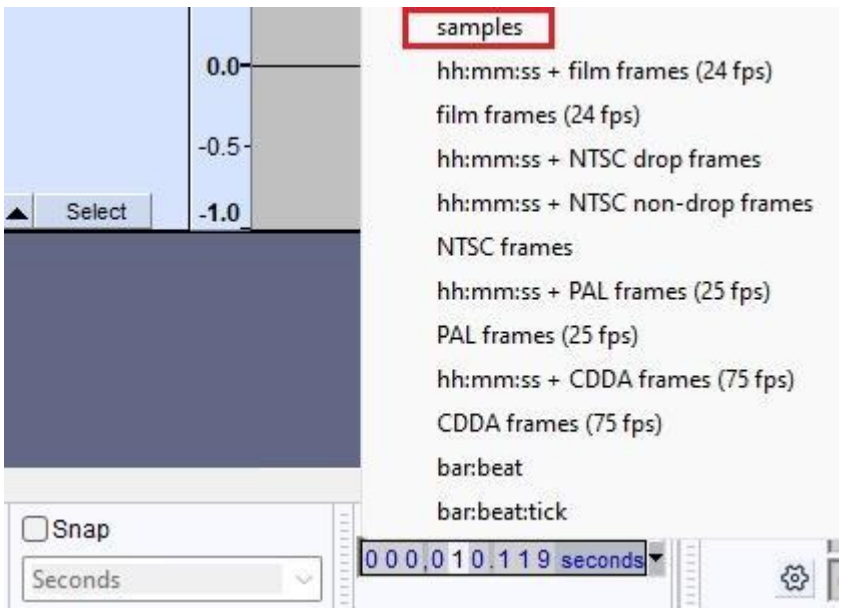
This section is entirely optional. Reading about this will clue you into what other FFXIV-adjacent programs like FFXIV Data Explorer or Voice Pack Creator mean by "samples", and will also let you set optional metadata for your audio within Audacity.

> Heads up: This metadata is purely for **OGG files**. If you're exporting as **WAV**, it most likely won't be there afterwards, or at least not detected by VFXEditor.

If you've ever inspected an OGG exported from an SCD, you may have noticed that two metadata tags exist under ***Edit > Metadata Editor***: LoopStart and LoopEnd. The numbers provided in those fields are represented by the start and end positions of the loop *in samples*, not milliseconds. In FFXIV's proprietary software, this sample data is likely used so that it can accurately and easily create a loop during the conversion process.

"*Well, what the heck are samples?*" you may be asking. To put it in simple terms, it's determined by those frequency values you see in audio tags, like 44.1kHz and 44kHz, which would be 44,100 and 44,000 samples per second, respectively. For example, with a file that's **44.1kHz**, at **20 seconds** the sample value would be **44100** samples *** 20** seconds, or **882000 samples**.

Even simpler, just tell your audio editor to display total track time in samples. In Audacity, this is found at the bottom of the window, when you click the arrow▾next to the marker time:



(The location of this may differ slightly if you have a different version or have customised it.) The other time values (labelled **Selection** in the newer Audacity version, or **Audio Selection/Position** in the old one) can also be modified to show samples.

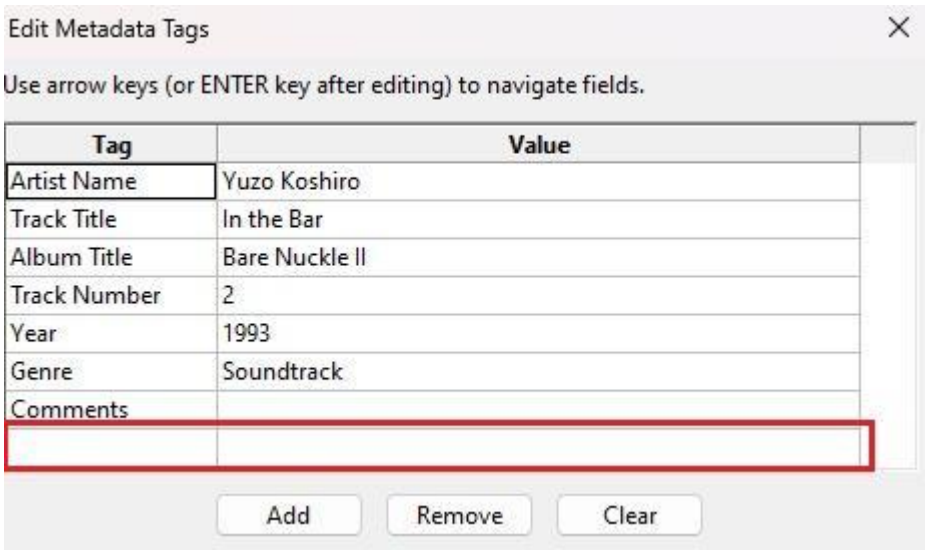
Now that we have the information and the view set up, we can edit the file's metadata in ***Edit > Metadata Editor*** and add in two new fields - **LoopStart** and **LoopEnd** - with the appropriate sample numbers at the start and end loop points that you marked with labels. This will be detected by VFXEditor and it will automatically set some values for you.

Apart from that, this metadata won't have significant impact, if any. However, on the off-chance that, in a future update, the sample values can be read *directly*, this will save time. Or maybe you're like me and just have it because it'd feel weird without it and because SE does it themselves. That's cool, too. Welcome to the weird club.

Some games read a LoopLength field also, but for FFXIV this is set in the .scd container manually.

> Small note: Interestingly, all original FFXIV SCD files imported - after doing a conversion to OGG - have their loops end where the music ends *except for exactly one additional sample afterwards*. I imagine this is a buffer of some sort. You may choose to do the same, but I find it makes little difference in the grand scheme of things.

Double click a blank field to type into it, and click **Add** to add more fields if necessary.



Preparing audio with FFmpeg

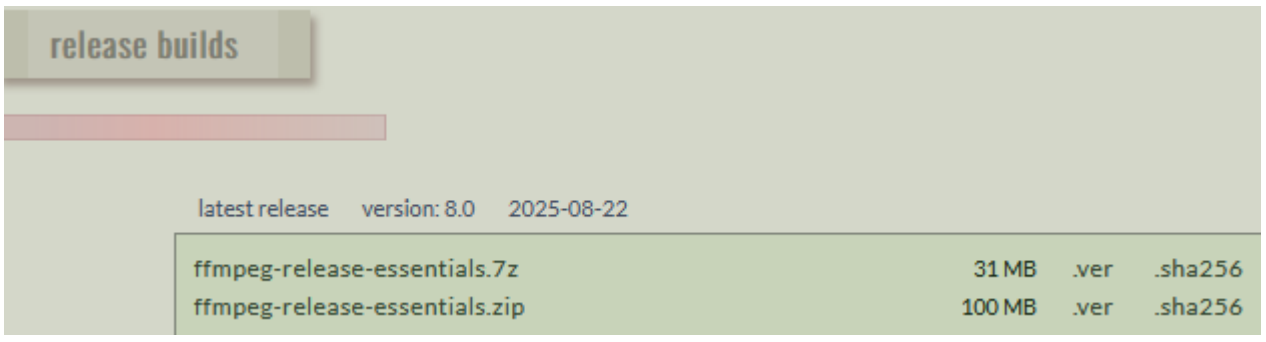
(Credit to **floppy drive** on the Students of Baldesion Discord for providing this method.)

This method is best used to tackle the looping system in OGG files, although it can additionally be used for basic file type conversions, outside of SCD.

You may wish to refer back to the Audacity preparation section if you haven't already so you can edit your audio as necessary before continuing. If you do so, make sure your saved audio is in a **lossless** format and **not OGG**, since we'll do that ourselves in a bit.

If you're willing to get your hands a little dirty with Windows' Command Manager/Terminal, there's an alternate method for establishing loops in your OGG-based audio, which has proven to be much, much nicer than the default OGG behaviour. I don't normally work with Linux environments, so I can't say at current how to do the right commands there, but I'll look into it if it's any different, you sudo users, you.

For this, you'll need to grab FFmpeg, a standalone codec. Normally, it's only available in source code format, but you can find current compiled builds here. You can pick either from the *git master builds* or the *release builds*; both are fine, though *git master builds* are more like alpha and may be slightly unstable in some aspects. Addtionally, you'll only need the one marked *essentials* for the purposes of this guide.



(**7z** and **zip** are the same files. **7z** just has better compression, but needs 7-Zip or a similar program installed to extract it.)

Extract the archive somewhere once you have it.

You'll also want a **lossless** sound file to work with - this means something like **WAV**, **FLAC**, or **ALAC**. As of right now, VFXEditor *cannot* handle what FFmpeg does with **MP3** to **OGG** and will crash your game if you try to import it. I'll show how to work around that in FFmpeg, but making it lossless in other programs is also acceptable and, more or less, the same result. Do what makes you comfortable. (You'll still ideally touch base with Audacity to find your loop points (easier), so keep that in mind.)

Open up the FFmpeg folder, then *bin*. Drop your audio file in this folder. Right click anywhere in this folder to open up a context menu, then either click *Open in Terminal* or *Command Prompt Here*.

If you wanted to do a conversion to lossless here instead of another program, the command is relatively simple: **ffmpeg -i *[file to convert]* *[converted file]***

You'd type it in there or **right click to paste** if you copy it from elsewhere. Press **Enter** to execute the command. (If you're working in Powershell, you'll want to make it **.\ffmpeg** instead because of access permissions.)

In this example, we're taking **nyancat.mp3** and converting it to **nyancat.wav**:

```
L:\DLs\ffmpeg\bin>ffmpeg -i nyancat.mp3 nyancat.wav
```

You should now have the converted file in the same folder.

For conversion from **lossless** to **OGG**, the command is a little more involved: **ffmpeg -i *[lossless file to convert]* -b:a *[bitrate]* -page_duration 1 *[converted file name]*.ogg**

In choosing a bitrate, you may wish to read the section on file sizes for reference. If you want a quick-and-dirty answer, **160k** is roughly equivalent to **192k MP3** files and is the recommended bitrate for general-use audio.

> The more you know: ***-i*** in these commands establishes that you're doing a conversion. **-b** means you're specifying a bitrate, while the **:a** is specifying all available audio streams (you can specify further by appending ***:[number of the stream]***, but for standard audio you only have one). **-page_duration** is what makes this particular tool useful for us. Even though **1** is technically massively overkill and you could equally use anything below about **20000** to roughly the same effect, this will ensure that we're no longer using the default paging setup that makes loops awful to work with in the first place. If you wish to read more about FFmpeg and the extent of its codec, and have five hours on your hands, see their official documentation here and information on commands for specific formats here.

Optional: you can also add LoopStart and LoopEnd metadata to your OGG by inserting **-metadata:s:0 LoopStart=*[sample value here]*** and **-metadata:s:0 LoopEnd=*[sample value here]*** right after your ***[lossless file to convert]***. (It can be anywhere in between file names, really, as long as it doesn't run into another command.) As this still needs to be manually tweaked in the editor as of September 2025, this step can be skipped.

For our example:

```
L:\DLs\ffmpeg\bin>ffmpeg -i nyancat.wav -b:a 160k -page_duration 1 nyancat.ogg
```

Or with the optional metadata addition:

```
L:\DLs\ffmpeg\bin>ffmpeg -i nyancat.wav -metadata:s:0 LoopStart=0 -metadata:s:0 LoopEnd=56131 -b:a 160k
 -page_duration 1 nyancat.ogg
```

You'll now have the specified **OGG** file in the *bin* folder, which you can safely move for use with VFXEditor, if you wish.

Note that you'll still need to specify any loop times through VFXEditor's interface. They will be second + millisecond values. However, since this method is somewhat unorthodox, any values you input for now will automatically round to the nearest hundredth of a millisecond (e.g. if you put in **6.144**, it will shift to **6.100**). Yes, we're still playing the numbers game, but they're more *manageable* numbers.

See the section on preparing audio with Audacity for finding loop points if need be.

File sizes, other conversion methods, and alternatives

This is an optional read. This section covers the disk size differences of relevant sound files methods of conversion to SCD outside of VFXEditor, and a small mention about Sebastina's Artemis Roleplaying Kit (ARK) and its use of MP3 files instead of SCD.

Comparison of sizes:

03 Morning Glow_signed_16bit_pcm.wav	9.53 MB	Wave Sound
03 Morning Glow_unsigned_8bit_pcm.wav	5.19 MB	Wave Sound
03 Morning Glow_ima_adpcm.wav	2.60 MB	Wave Sound
03 Morning Glow_quality10_48k.ogg	2.53 MB	OGG Video File (VLC)
03 Morning Glow_quality10_44.1k.ogg	2.50 MB	OGG Video File (VLC)
03 Morning Glow_microsoft_adpcm.wav	2.40 MB	Wave Sound
03 Morning Glow_cbr (constant bit rate)_320k.mp3	2.16 MB	MP3 Format Sound
03 Morning Glow_vbr (variable bit rate)_320k.mp3	2.08 MB	MP3 Format Sound
03 Morning Glow_quality8_44.1k.ogg	1.46 MB	OGG Video File (VLC)

> Note: While Microsoft ADPCM (sometimes referred to as 4-bit ADPCM), IMA ADPCM, and MP3 are listed here, these are currently not supported for import into VFXEditor. They're simply for general comparisons.

It is not recommended to use unsigned 8-bit PCM. This will add a noticeable degree of background hiss/static to your audio, especially during quieter sections.

There's also 24-bit and 32-bit PCM that I didn't list here (they'd be 14.3MB and 19MB, respectively), but there's zero reason to use these in FFXIV unless you hate your storage space. The quality gain is extremely negligible for everyday use. But I don't pay your sub, so if you want to use these encodings, that's entirely your decision.

(Additionally, XMA and ATRAC are technically supported by SCD internally, but the editor is not currently equipped to read these types.)

For quality reference on OGG files:

160kbps = 5	192kbps = 6
224kbps = 7	256kbps = 8
320kbps = 9	500kbps = 10

In other words, a quality of 10 will essentially supersample a standard MP3 file. A regular user with $20 headphones from Walmart will normally not be able to discern too much of a difference, if any, between 192 kbps/quality 6 and 500 kbps/quality 10. The major reason to consider the higher qualities is to improve your own ability to set custom loop points. Even then, you can get away with lower qualities, assuming you're fine with spending a bit more time on finicking with loop times and that the source audio doesn't immediately fade right after the point it should loop.

Should you not be worried about loops outside of start and end, setting your OGG export to a quality of 6 is perfectly fine.

While Audacity is my go-to for edits, you may not need all that functionality and only want to turn your audio into something you can import into VFXEditor and be done with it. If that's the case, you have a few options:

- You can use an online converter. It's quick and does the job, albeit with slightly less control over how it does all that. There are plenty of sites if you just search for something like "mp3 to ogg" or "mp3 to wav" or whatever file type you're converting from. Some files converted this way can cause issues with playback or crashing, so either try another site or a dedicated program to see if the problem persists.

- If you have VLC installed, go to *Media > Convert/Save*. Select your file with *+ Add...* (or drag and drop), then click the *Convert/Save* button at the bottom. Next to *Profile* in the window that pops up, you can select their default *Audio - Vorbis (OGG)* profile (it's about a quality of 4), or click either the wrench button to modify that or the third button to make a custom profile. *Encapsulation* is just the file extension - so, this would either be *Ogg/Ogm* or *WAV*. Make sure, in the Audio Codec tab, that *Audio* is checked. Pick either *Vorbis* for OGG or *WAV* for, well, WAV. Specify your desired bitrate, channels (2 for stereo, 1 for mono), and sample rate (44.1k is standard, but double check your original file's properties so that it matches). Click the *Create* (*Save*) button once you're done, scroll in the *Profile* dropdown until you find the it, select it, choose your *Destination*, and click *Start*.

- Foobar2000 also has conversion available and is more precise in its quality values for OGG. Drag and drop your audio in the program (or select it if it's there), right click it, then go to *Convert > ...* . You can click on *Output Format* on the right-hand side of the window to select your format. For WAV, the Auto *Output bit depth* will be 16-bit. For OGG, you can click *Edit* to modify its quality. Once done, click *OK,* then *Back*, and then finally *Convert*.

As stated elsewhere, you can also take an MP3 and convert it directly to SCD with programs like FFXIV Data Explorer (needs JRE/JDK 11) or Voice Pack Creator. That said, FFXIV Data Explorer doesn't always work or will do odd things like pitch shifting. Voice Pack Creator is fine for sound effects or voices, but for BGM this is not a software I can really point others to yet because of how it handles SCD creation. Loops will also not work properly with it. Both additionally use sample numbers for their BGM conversions, which requires some effort to find out, and even then it may not work as it should.

Lastly, from the same creator as Voice Pack Creator - Sebastina - there's the Artemis Roleplaying Kit plugin. This is mostly for taking your MP3s and replacing existing sound effects or voices with them. (I don't entirely know if it affects more than those.) It also comes with some additional features like playing Twitch stream audio, adjusting specific volumes that aren't covered by FFXIV's default volume sliders, or changing dance mod audio from the BGM channel to the performance channel.

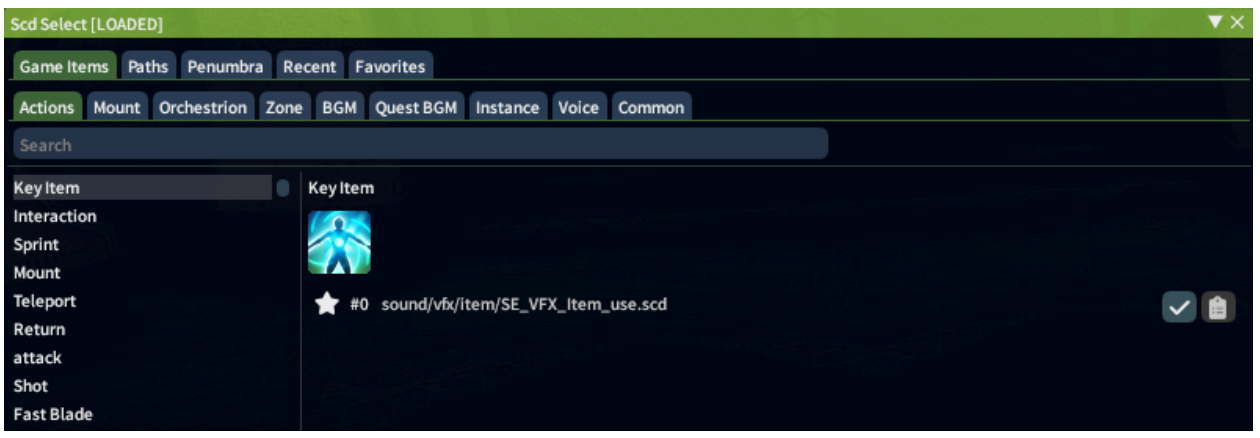
A mini guide to finding in-game sounds

This is an optional read.

Maybe you wanted to edit an existing SCD pulled directly from the game. Or maybe you've always heard something playing in the background and thought it was just the coolest thing, wanted to use it, but got frustrated with finding it. Or perhaps there's a sound you want to check from an installed mod. Whatever the case may be, go ahead and open up the VFXEditor plugin with FFXIV running. You can use the chat command **/vfxedit scd** to open it right to the *SCD Editor*. (If you're using the beta version in its own custom repository, this command will be **/vfxbeta scd** instead). Or you can also find it directly in the *Plugin Installer* window provided by Dalamud.

With the SCD window open, click on the magnifying glass to the right of *Loaded SCD*:

Here, you have direct access to a vast number of SCDs that can be found in the game's index files, in loaded Penumbra mods, or in any local folder. You're able to do searches by string, by category, or even get it directly, provided you know its file path in the index.

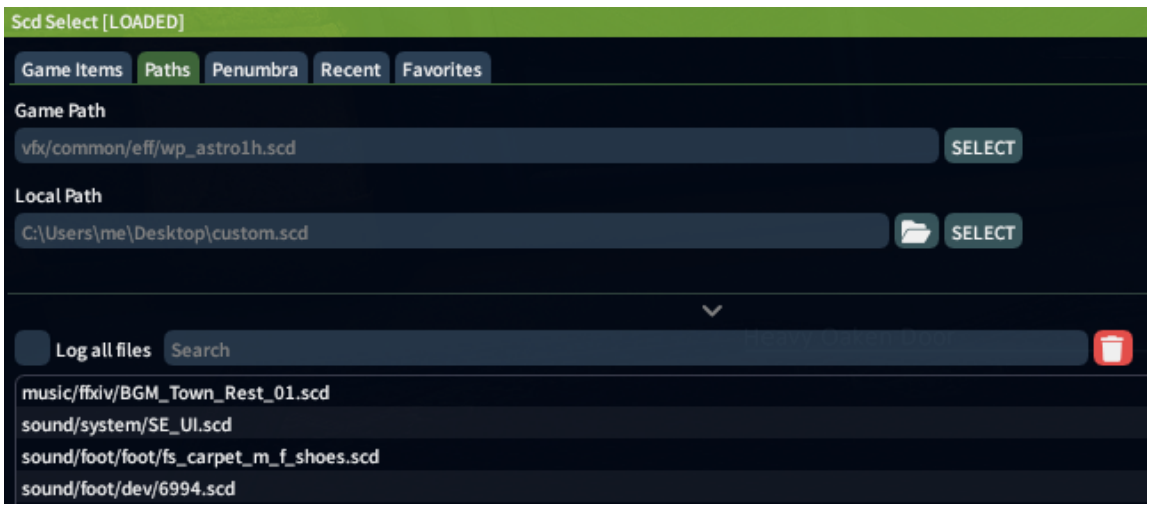


Let's go back to the main SCD Editor window. Go to ***File > Tools***.

Here we have the *Tools* window with the *Resources* tab. This shows you every asset in use in the current area, updated every ten seconds, along with its path. Opening up *Sound > scd* in this window here will show any sound effects. Likewise, *Music > scd* will show the currently playing music.

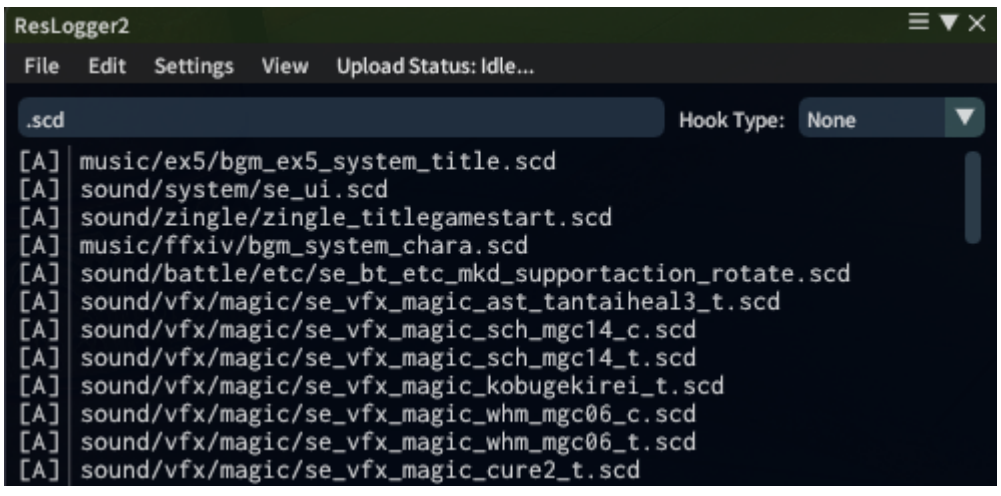
You can even do exports any raw files from the game's index in the *Utilities* tab, whether it be sounds, VFX, models, textures, etc. If it exists natively within the game, you can extract it here, provided you have the path to it.

Additionally, if you navigate to the *Paths* tab back in the *SCD Select* window, you'll notice you can log what's currently being used and played:



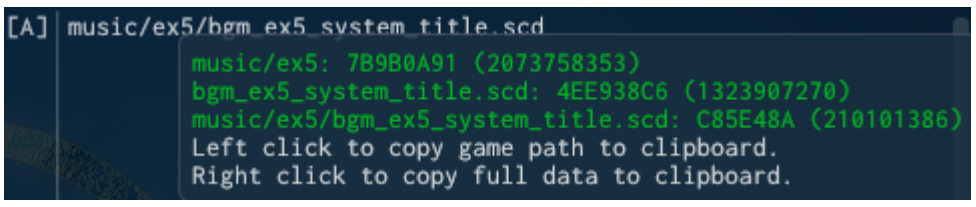
The problem here is that it logs **everything**, no matter how many times a single sound's been used in the span of a second. So, if you're looking for a specific sound in a busy place, you may not find this ideal, as you'll have a billion sounds zooming past. You also can't click to copy, if you need the file path. But it's here if you need it. There's also a similar resource logger built into Penumbra, but it suffers from a lot of the same issues, as well as degrading your game's performance while it runs.

This is where the nifty **ResLogger2 plugin** comes into play. It will pick up the files being used in real time and is significantly easier to parse through. It also prevents duplicate entries from showing if you click ***Settings > Log unique paths only*** in the plugin window. You can filter by search term (like **.scd** to restrict it to only sounds) like so:



In order to see custom paths (ones that don't exist in the game's index, but are used by mods), you'll need to enable that in ***Settings > Log paths that don't exist***.

Additionally, while this plugin actually lets you copy file paths, this also needs to be enabled at ***Settings > Show hash tooltip***. This will give you a mini window that shows up when you hover over a path, but more importantly you can now left or right click to copy what's there. For 99% of cases, just left-click to grab only the path; otherwise, you'll get a lot of excess data.



Lastly, you can copy all unique paths currently in the log so you can paste it elsewhere, by going to ***Edit > Copy unique paths***. Just a general warning that, since it copies *every* asset, your output log can potentially be humongous.

The **Sound Filter plugin** is also a good option, as it pulls sounds along with each called index number every time something plays. However, it's a nightmare if you're not in an isolated spot or being careful, since (the same as the logging in VFXEditor or Penumbra) even a mini's every footstep and UI hover sound are recorded as individual entries.

Another resource I neglected to mention is the **standalone .exe Anamnesis**. Its main purpose is to preview things like weapons, hairstyles, skin/hair colour, dyes, and so on in a convenient fashion or for use with /gpose. By extension, any associated sounds with regards to special weapon skins or a change in voice will be reflected in-game (purely client side, so it's safe), making it easier to pinpoint certain sounds without actually having the requirement. Minions and mounts won't really change their sounds, unfortunately, although dismissing a mount will play its unique sound. The **Glamourer plugin** has a similar function, but the only sounds that get modified there (as far as I know) are weapons.

Additionally, the **Orchestrion plugin** can be pulled in for music searches if you know its general content location or official name.

Or you can check out my in-progress .scd list. Naturally, it won't have everything, and descriptions of sounds are somewhat sparse, but you may find what you're looking for or even see something that could spark inspiration.

Working with SCD by examples

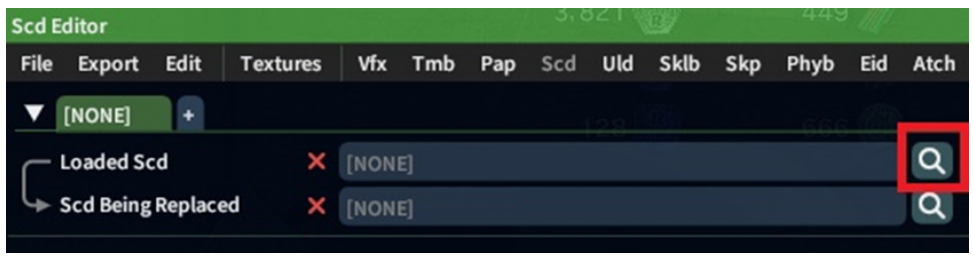
- Basic **looped** BGM swap for an in-game BGM
- Basic **non-looped** BGM swap for an in-game zone
- Custom four-channel dynamic BGM swap for an in-game dynamic BGM
- Placing a (looped) custom song on a (looped) emote that doesn't have one
- Adding a sound to a skill, animation, or VFX
- Replacing an existing modded sound with your own
- Cropping sounds to only play specific sections
- Enabling the use of randomised BGM/SFX for animations/skills

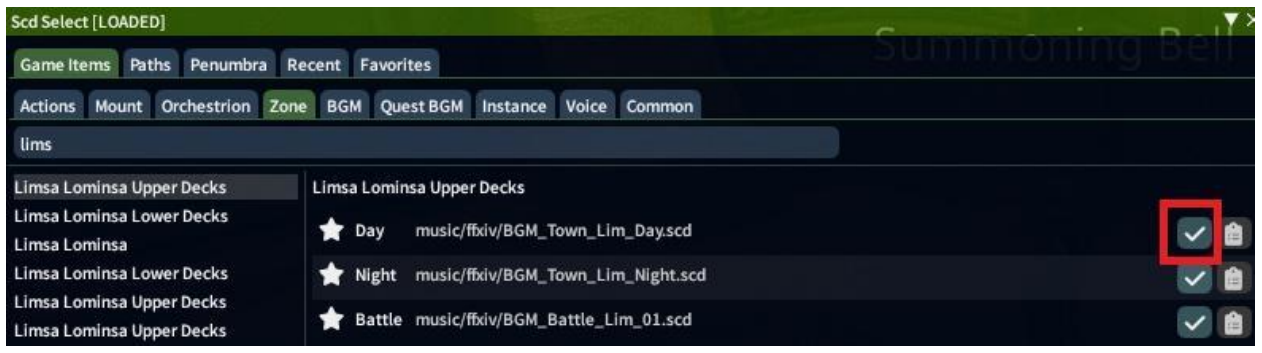
Basic looped BGM swap for an in-game BGM, using a prepared song

Note: If you're just looking to swap vanilla BGM (except orchestrion rolls), this can be done through the Orchestrion plugin, no VFXEditor required.

Start by picking the SCD that you want to use as a template. In this example, I'm choosing the Limsa day theme BGM, as it natively includes looping.

Side note: You may not get intended results when you're replacing ARR (base game) dungeon or instanced music that doesn't loop or repeat by default. This is due to a technical limitation. However, replacing music that repeats after some silence (such as open-area zones like Western Thanalan or near aetherytes) should work just fine.





You can additionally search through the Orchestrion plugin if need be.

Click on *Audio 0* back in the main window once you've selected the BGM, so that we can work with the loop points and replacements.

Your song ideally should be in **OGG** format. It can be **WAV**, but it will auto-convert to OGG when you import it.

If you have an **.mp3/.flac/.m4a** (etc.), convert it to **OGG** (or **WAV**) with any software or site you choose before you import it.

Side note: HCA files are also being introduced into SCDs as we head into the Evercold expansion. Support for this is still ongoing and may not produce the correct results with loops, so if you see this encoding shown below the mini player, it's recommended to find one that has OGG for now. I'll update this note once it stabilises.

Now look for the rightmost button next to VFXEditor's built-in player and click it to begin searching for your song.

Navigate to where you have your song saved, select it, and click *Ok*.

Audio 0 will have closed itself to refresh the data; click it open again.

The left number is your loop's starting time, while the right is the loop's ending time. Both are measured in seconds. They are different for every unique OGG/WAV. If you choose to edit these, you may see your inputs change a bit after you type them in. Don't worry - this is normal. Due to how OGG files work with something called pages, the best that can be done is an *approximation of time*. The higher quality your OGG export is, the better this approximation ends up being. That means we'll still have to put in some elbow grease if we want to fine tune our loops, but we know where to start.

If you did nothing special with your OGG/WAV and it doesn't have any relevant *LoopStart* or *LoopEnd* metadata, the left one will be **0.000** because it's starting from the beginning of your song, while the right one should automatically be set to the end of your song. If the right value is also **0.000**, type in an absurdly long value (like 9999999999) and it'll auto-adjust to be the correct ending time. If the file was **WAV**, it may not automatically add the end time, and it won't auto-adjust if you type one in, so you'll have to look for it (in Audacity or the mini player in VFXEditor itself) and then add it in. Assuming you only want to loop your song from start to finish, you don't have to do anything else here.

If you skipped past the section on working with Audacity and want to make custom loop points, you'll probably want to check against the built-in player for some values to start with. Enable the **Simulate Loop Start/End** setting in the *Settings* dropdown to make this a little easier.

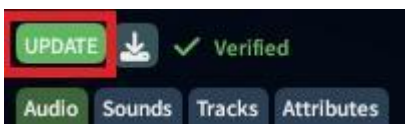
Click around to find some acceptable spots to start and end your loop; type those into the **Loop Time** fields - left for start, right for end. When you put the numbers in, the player will draw green and red lines to mark those points so you can see them better.

If you followed the Audacity preparation at the start of this guide, it's likely you'll want to fine tune the numbers that were filled out for you if your file is **OGG**.

Once you've done whatever you need to do with the *Loop Time* fields, let's pick the BGM being replaced.

For our example, let's say we're replacing the Gridania day theme. We'll use the same method as before to find it. It'll show up as **[MUSIC] ffxiv/BGM_Town_Gri_Day** once you select it.

Now we're at the testing phase. In order to have your song be heard in-game, the simplest method is to click the UPDATE button. It'll be above the *Audio* tab.



Important**: if you're not using the Orchestrion plugin to do your tests, do not be in the same zone the replaced BGM affects when you click UPDATE or you will crash when it tries to loop.**

If you're using the Orchestrion plugin (highly recommended), you have two methods available to avoid original BGM crashing you on update.

- Click to play two BGM that are completely different from the one you're modifying. Click UPDATE in VFXEditor. Then click *Stop* if you're in the right zone, or click the zone's BGM you want to test.

- Click the **1 - None** entry. Wait about five seconds for the BGM to completely fade out. Click UPDATE in VFXEditor. Then click *Stop* if you're in the right zone or click the zone's BGM you want to test.

For testing, if you're not using the Orchestrion plugin, there are three methods to be sure the original BGM doesn't start playing again once you return to the affected zone.

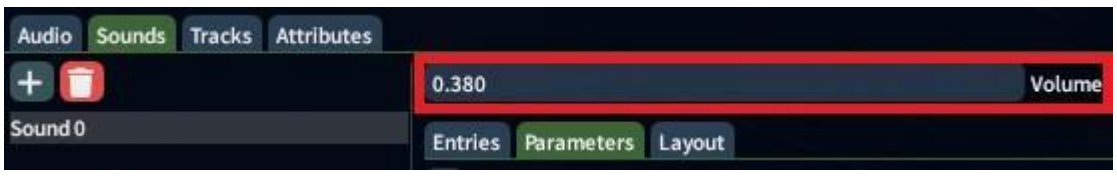
- Teleport to two different areas first, click UPDATE, and then go back. Not recommended because it incurs a lot of gil cost, and some of us are living off the streets. But if you've got the money to burn, sure, you do you.

- Teleport somewhere else, click UPDATE, wait about 30 seconds, and teleport back. Also not recommended because you'll spend a lot of time twiddling your thumbs, but it's great if you have things to do on the side.

- Queue up for a solo unsynced or Explorer Mode duty, hop in there, click UPDATE, immediately leave. Takes maybe 10 seconds.

If the loop still isn't right, adjust the **Loop Time** values as necessary, then **repeat the testing step you used above to ensure you don't crash**. Yes, it's absolutely a hassle, but unfortunately there's no quick and simple way to do this yet. Maybe someday.

If you think the volume is too loud or soft, you can edit it in the *Sounds* tab.

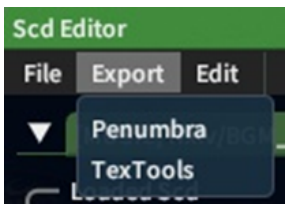


This usually can only be between 0.000 and 1.000 - it normally won't respond to anything higher or lower than that range for BGM. If it still needs major adjustment beyond that, you'll have to manually edit the volume gain outside of VFXEditor. (There *is* a way to manipulate it, but it also changes how the sound plays, so I don't really recommend it unless you're desperate. Look for **ADSR** in this doc if you're curious.)

Side note: The **Loop** checkbox in ***Sounds > Sound # > Parameters*** helps to flag the sound for looping. There's additionally a **Music** checkbox, but contrary to what you might think, checking this box does nothing. Really. The **Bus Number** field in itself is what classifies it in-game as music or not. You're free to check it if you really want to. It doesn't hurt. But it still doesn't do anything. You do you, boo.

If your loop isn't being affected at all by **Loop Time** values or is doing strange things, I would additionally look into changing the **Interval** items in the *Tracks* tab. For looped BGM, the first **Interval** is the loop start time in milliseconds and the second **Interval** is the length of the loop in milliseconds.

Once you're satisfied, you're basically done, apart from exporting as a mod or what have you.



Even if it sounded fine in the test run, I would still take the time to enable the mod yourself and see if anything needs work in the actual environment. This can be especially true when it's gone through several loops of the song, as it may hiccup on the second or third time, or if the song doesn't feel balanced during combat.

While the game is running, VFXEditor keeps your (temporary) changes upon clicking UPDATE. Therefore, you'll likely want to revert the BGM to its vanilla state once you're done working with it. There are a couple ways to do this:

- ***File > New***. This wipes your currently open window of all edits. For sounds, this should be all you need to do.

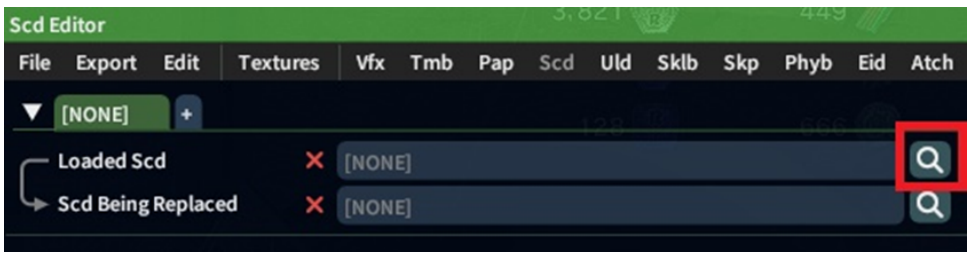
- In case the changes are being sticky: for the **Scd Being Replaced** field, choose the BGM you picked for **Loaded Scd** and click UPDATE, then click the **X** next to each field to unload them.

- In case it's *still* not cooperating, log out to character select and back in. You can also restart your game, as VFXEditor will not keep any changes after the game's been closed. Should something *still* be wrong, check that another plugin or program isn't interfering.

Basic non-looped BGM swap for an in-game zone, using a prepared song

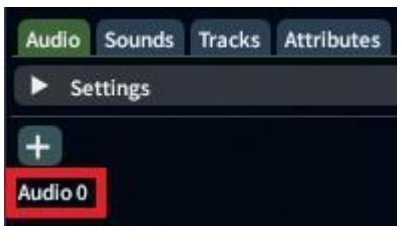
Note: If you're just looking to swap vanilla BGM (except orchestrion rolls), this can be done through the Orchestrion plugin, no VFXEditor required.

Start by picking the SCD that you want to use as a template. In this example, I'm choosing a Thanalan field theme BGM - **music/ffxiv/BGM_Field_Uru_06.scd** - as it natively works without loops.



You can additionally search through the Orchestrion plugin if need be.

Click on *Audio 0* back in the main window once you've selected the BGM.



Your song ideally should be in **OGG** format. It can be **WAV**, but it will auto-convert to OGG when you import it.

If you have an **.mp3/.flac/.m4a** (etc.), convert it to **OGG** (or **WAV**) with any software or site you choose before you import it.

> Side note: HCA files are also being introduced into SCDs as we head into the Evercold expansion. Support for this is still ongoing and may not produce the correct results with loops, so if you see this encoding shown below the mini player, it's recommended to find one that has OGG for now. I'll update this note once it stabilises.

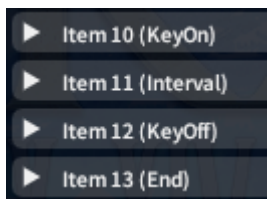
Now look for the rightmost button next to VFXEditor's built-in player and click it to begin searching for your song.

Navigate to where you have your song saved, select it, and click *Ok*.

Audio 0 will have closed itself to refresh the data; click it open again.

Make sure **both *Loop Time*** fields read **0.000**. If they don't, be sure to change them.

Now we want to take a detour to the *Tracks* tab and click on *Track 0*. You'll see a series of items, and the ones we want are towards the bottom:



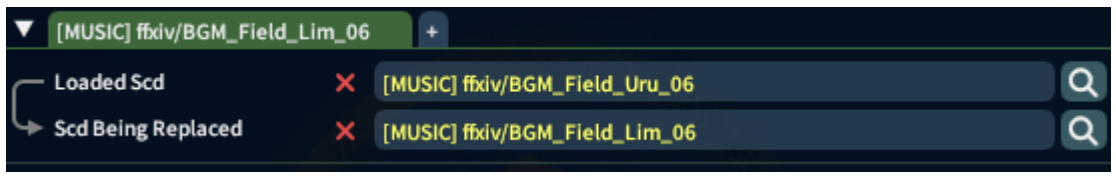
Sometimes the item numbers won't be exactly the same, and that's okay - as long as they're in this order. Specifically, though, we're interested in the **Interval** item. Click on it to open it up.

This *Value* is the **length of your BGM in milliseconds**. We want to change it to match our custom sound's length (found through the built-in player in VFXEditor by scrolling to the end of the sound when played, or a program like Audacity). What this field does is ensure that your sound doesn't get cut short or clip, and that it doesn't affect how it plays if there's a forced repeat (not loop) of the BGM.

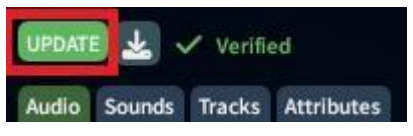
Additionally, under ***Sounds > Sound 0 > Parameters***, the checkbox for **Loop** should be unchecked, if it's not already.

Once you've done whatever you need to do with all of that, let's pick the BGM being replaced.

For our example, let's say we're replacing a Limsa field theme BGM. We'll use the same method as before to find it and select it.



Now we're at the testing phase. In order to have your song be heard in-game, the simplest method is to click the UPDATE button. It'll be above the *Audio* tab.



> **Important: if you're not using the Orchestrion plugin to do your tests, do not be in the same zone the replaced BGM affects when you click UPDATE or you might crash when it tries to access it again.**

If you're using the Orchestrion plugin (highly recommended), you have two methods available to avoid original BGM crashing you on update.

- Click to play two BGM that are completely different from the one you're modifying. Click UPDATE in VFXEditor. Then click *Stop* if you're in the right zone, or click the zone's BGM you want to test.

- Click the **1 - None** entry. Wait about five seconds for the BGM to completely fade out. Click UPDATE in VFXEditor. Then click *Stop* if you're in the right zone or click the zone's BGM you want to test.

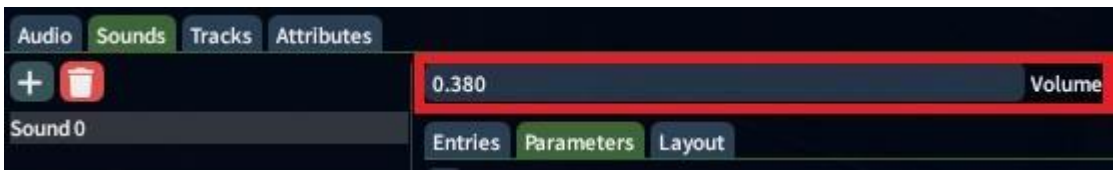
For testing, if you're not using the Orchestrion plugin, there are three methods to be sure the original BGM doesn't start playing again once you return to the affected zone.

- Teleport to two different areas first, click UPDATE, and then go back. Not recommended because it incurs a lot of gil cost, and some of us are living off the streets. But if you've got the money to burn, sure, you do you.

- Teleport somewhere else, click UPDATE, wait about 30 seconds, and teleport back. Also not recommended because you'll spend a lot of time twiddling your thumbs, but it's great if you have things to do on the side.

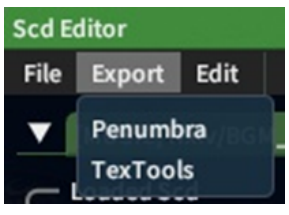
- Queue up for a solo unsynced or Explorer Mode duty, hop in there, click UPDATE, immediately leave. Takes maybe 10 seconds.

If you think the volume is too loud or soft, you can edit it in the *Sounds* tab.



This usually can only be between 0.000 and 1.000 - it normally won't respond to anything higher or lower than that range for BGM. If it still needs major adjustment beyond that, you'll have to manually edit the volume gain outside of VFXEditor. (There *is* a way to manipulate it, but it also changes how the sound plays, so I don't really recommend it unless you're desperate. Look for **ADSR** in this doc if you're curious.)

Once you're satisfied, you're basically done, apart from exporting as a mod or what have you.



Even if it sounded fine in the test run, I would still take the time to enable the mod yourself and see if anything needs work in the actual environment. This can be especially true when it's gone through a repeat of the song, as it may hiccup then, or if the volume doesn't feel balanced overall.

While the game is running, VFXEditor keeps your (temporary) changes upon clicking UPDATE. Therefore, you'll likely want to revert the BGM to its vanilla state once you're done working with it. There are a couple ways to do this:

- ***File > New***. This wipes your currently open window of all edits. For sounds, this should be all you need to do.

- In case the changes are being sticky: for the **Scd Being Replaced** field, choose the BGM you picked for **Loaded Scd** and click UPDATE, then click the **X** next to each field to unload them.

- In case it's *still* not cooperating, log out to character select and back in. You can also restart your game, as VFXEditor will not keep any changes after the game's been closed. Should something *still* be wrong, check that another plugin or program isn't interfering.

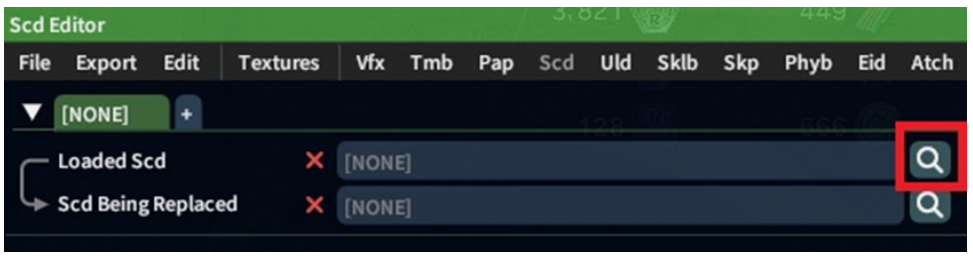
Custom four-channel dynamic BGM swap for an in-game dynamic BGM

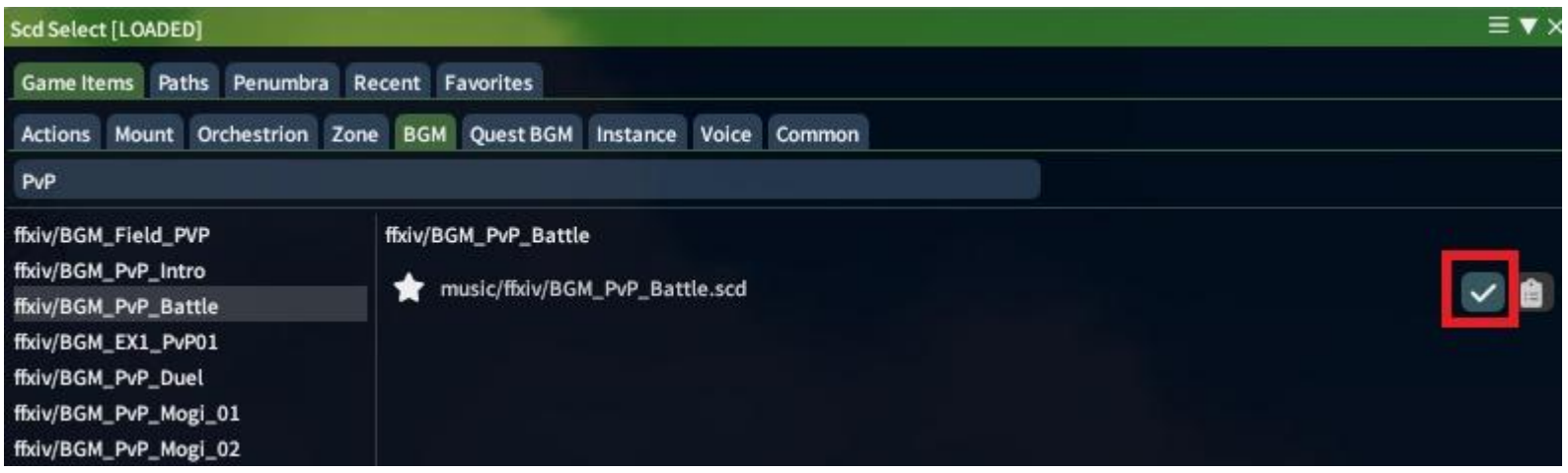
For use with BGM that detects combat status - like PvP, some alliances, the time stop mechanic in A12, and instanced operations like Bozja/Eureka - specifically those with the ***DynamixStream*** Type under ***Sounds > Sound # > Parameters***. Note that combat in most open areas and some later alliances do not utilise this and are standard BGM.

Note: If you're just looking to swap vanilla BGM (except orchestrion rolls), this can be done through the Orchestrion plugin, no VFXEditor required.

Important: If you haven't gone through the corresponding section in Audacity on this topic, head over there now. Unless your audio already contains four channels, you will need to manually set it up through Audacity or a similar external program.

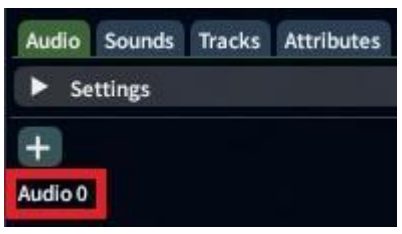
Start by picking the SCD that you want to use as a template. In this example, I'm choosing the main PvP battle theme.





You can additionally search through the Orchestrion plugin if need be.

Click on *Audio 0* back in the main window once you've selected the BGM, so that we can work with the loop points and replacements.



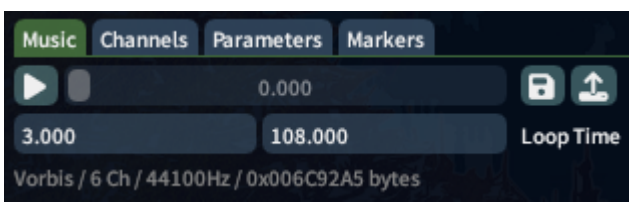
Your song ideally should be in **OGG** format. It can be **WAV**, but it will auto-convert to OGG when you import it.

> Side note: HCA files are also being introduced into SCDs as we head into the Evercold expansion. Support for this is still ongoing and may not produce the correct results with loops, so if you see this encoding shown below the mini player, it's recommended to find one that has OGG for now. I'll update this note once it stabilises.

Now look for the rightmost button next to VFXEditor's built-in player and click it to begin searching for your song.

Navigate to where you have your song saved, select it, and click *Ok*.

Audio 0 will have closed itself to refresh the data; click it open again.



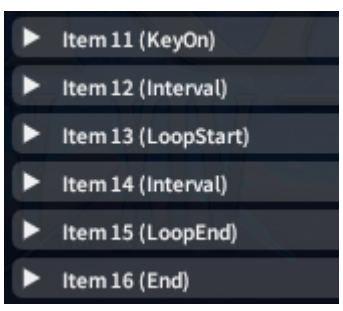
The left number is your loop's starting time, while the right is the loop's ending time. Both are measured in seconds. They are different for every unique OGG. If you choose to edit these, you may see your inputs change a bit after you type them in. Don't worry - this is normal. Due to how OGG files work with something called pages, the best that can be done is an *approximation of time*. The higher quality your OGG export is, the better this approximation ends up being. That means we'll still have to put in some elbow grease if we want to fine tune our loops, but we know where to start.

If you didn't add the optional *LoopStart* or *LoopEnd* metadata, the left one will be **0.000** because it's starting from the beginning of your song, while the right one should automatically be set to the end of your song. If the right value is also **0.000**, type in an absurdly long value (like 9999999999) and it'll auto-adjust to be the correct ending time. Assuming you only want to loop your song from start to finish, you don't have to do anything else in this tab.

If you did, it'll automatically set to the nearest approximation of each.

You'll probably want to fine tune the numbers that were filled out for you.

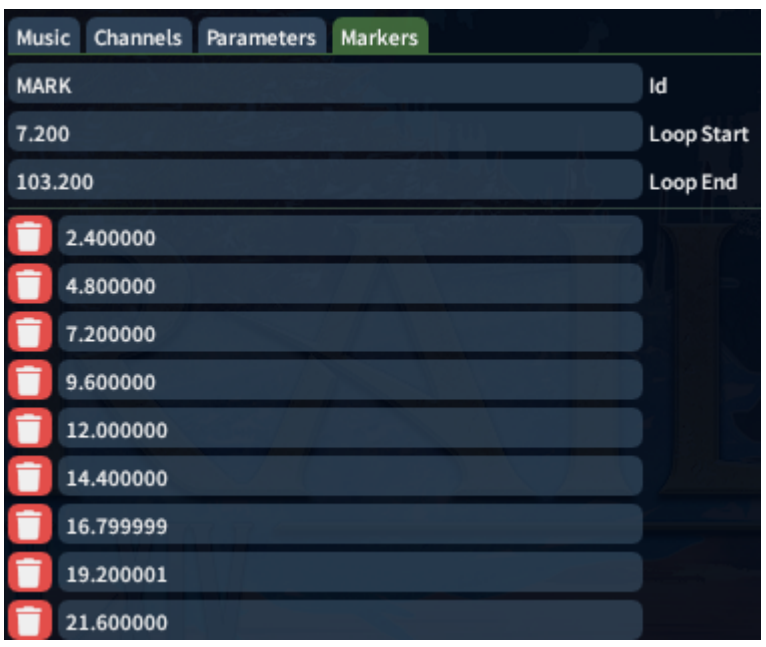
Next, we want to take a detour to the *Tracks* tab and click on *Track 0*. You'll see a series of items, and the ones we want are towards the bottom:



Sometimes the item numbers won't be exactly the same, and that's okay - as long as they're in this order. Specifically, though, we're interested in the **Interval** items. Click on them to open them up.

For looped BGM, the first **Interval** is the loop start time in milliseconds and the second **Interval** is the length of the loop in milliseconds. Make sure you change these to what you wrote down to ensure the loop goes off without a hitch.

Now, this is where all those labels and transition time values come into play. You should see a tab called *Markers* in here - click on it.

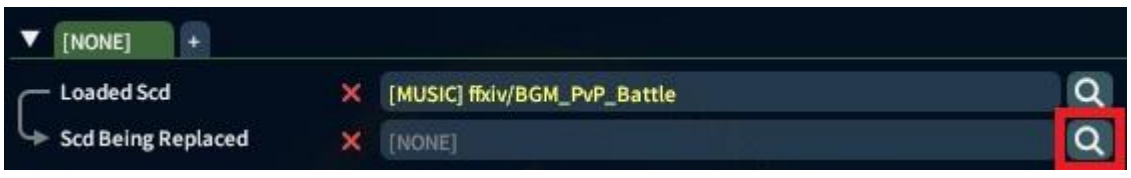


You've probably guessed it all by now, but for each time value you have written down, you want to plug them in here - one field for one unique time value, going in order. You can delete entries as necessary, or add more with the **+ New** button at the bottom of this tab. Make sure the **Id** field says **MARK**.

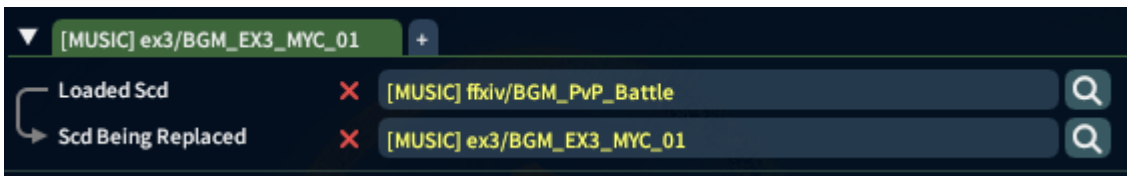
The **Loop Start** and **Loop End** fields generally should match what you set in the **Loop Time** fields from earlier, but doesn't necessarily have to - I need to verify this, but I have a feeling the markers' **Loop Start** sets where the BGM continues after it's been interrupted by something like boss BGM.

> Side note: In vanilla BGM, there's some general discrepancy between the last marker time set, the **Loop End** value, and the loop **Interval** value set in the *Tracks* tab. (If you don't know about this **Interval** value yet, we'll get to it.) I'm unsure why this is, as the whole premise of markers is very new to the editor, but it may just be failsafes or things that didn't end up being used. In any case, don't worry too much about it.

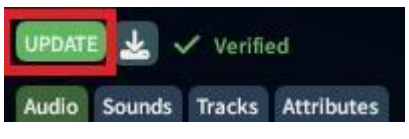
Once you've done whatever you need to do with all of that, let's pick the BGM being replaced. As stated before, this will only work properly if you replace files with a similar function, so you cannot pick something that doesn't natively support dynamic BGM.



For our example, let's say we're replacing the Bozja field theme. We'll use the same method as before to find it.



Now we're at the testing phase. In order to have your song be heard in-game, the simplest method is to click the UPDATE button. It'll be above the *Audio* tab.



> **Important: if you're not using the Orchestrion plugin to do your tests, do not be in the same zone the replaced BGM affects when you click UPDATE or you will crash when it tries to loop.**

If you're using the Orchestrion plugin (highly recommended), you have two methods available to avoid original BGM crashing you on update. Note that if you don't plan to test in the affected zone, you'll want to go into Orchestrion's settings window and make sure ***Handle special "in-combat" and mount movement BGM modes*** is checked so that you can hit a target to trigger the combat status.

- Click to play two BGM that are completely different from the one you're modifying. Click UPDATE in VFXEditor. Then click *Stop* if you're in the right zone, or click the zone's BGM you want to test.

- Click the **1 - None** entry. Wait about five seconds for the BGM to completely fade out. Click UPDATE in VFXEditor. Then click *Stop* if you're in the right zone or click the zone's BGM you want to test.

For testing, if you're not using the Orchestrion plugin, there are three methods to be sure the original BGM doesn't start playing again once you return to the affected zone.

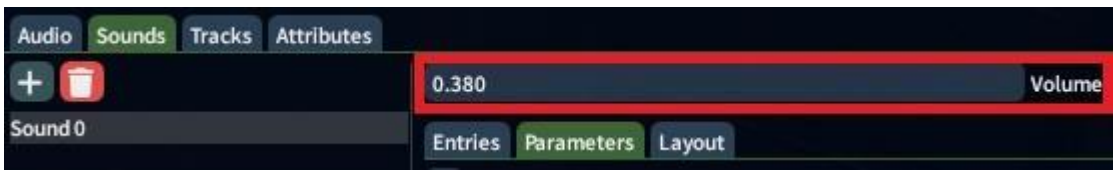
- Teleport to two different areas first, click UPDATE, and then go back. Not recommended because it incurs a lot of gil cost, and some of us are living off the streets. But if you've got the money to burn, sure, you do you.

- Teleport somewhere else, click UPDATE, wait about 30 seconds, and teleport back. Also not recommended because you'll spend a lot of time twiddling your thumbs, but it's great if you have things to do on the side.

- Queue up for a solo unsynced or Explorer Mode duty, hop in there, click UPDATE, immediately leave. Takes maybe 10 seconds.

If the loops and marker placements still aren't right, adjust the **Loop Time** values as necessary, then **repeat the testing step you used above to ensure you don't crash**. Yes, it's absolutely a hassle, especially for this type of BGM replacement, but unfortunately there's no quick and simple way to do this yet. Maybe someday.

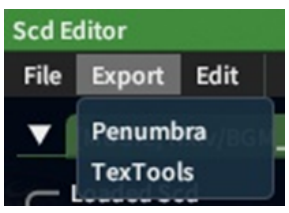
If you think the volume is too loud or soft, you can edit it in the *Sounds* tab.



This usually can only be between 0.000 and 1.000 - it normally won't respond to anything higher or lower than that range for BGM. If it still needs major adjustment beyond that, you'll have to manually edit the volume gain outside of VFXEditor. (There *is* a way to manipulate it, but it also changes how the sound plays, so I don't really recommend it unless you're desperate. Look for **ADSR** in this doc if you're curious.)

> Side note: The **Loop** checkbox in ***Sounds > Sound # > Parameters*** helps to flag the sound for looping. There's additionally a **Music** checkbox, but contrary to what you might think, checking this box does nothing. Really. The **Bus Number** field in itself is what classifies it in-game as music or not. You're free to check it if you really want to. It doesn't hurt. But it still doesn't do anything. You do you, boo.

Once you're satisfied, you're basically done, apart from exporting as a mod or what have you.



Even if it sounded fine in the test run, I would still take the time to enable the mod yourself and see if anything needs work in the actual environment. This can be especially true when it's gone through several loops of the song, as it may hiccup on the second or third time, or if the song doesn't feel balanced during combat.

While the game is running, VFXEditor keeps your (temporary) changes upon clicking UPDATE. Therefore, you'll likely want to revert the BGM to its vanilla state once you're done working with it. There are a couple ways to do this:

- ***File > New***. This wipes your currently open window of all edits. For sounds, this should be all you need to do.

- In case the changes are being sticky: for the **Scd Being Replaced** field, choose the BGM you picked for **Loaded Scd** and click UPDATE, then click the **X** next to each field to unload them.

- In case it's *still* not cooperating, log out to character select and back in. You can also restart your game, as VFXEditor will not keep any changes after the game's been closed. Should something *still* be wrong, check that another plugin or program isn't interfering.

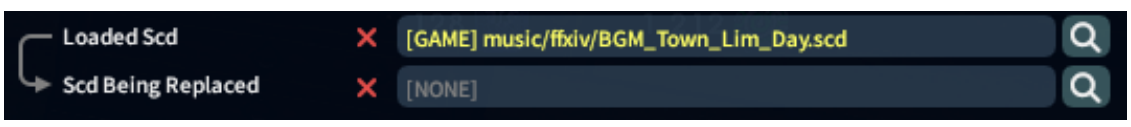
Placing a looped custom song on a (looped) emote that doesn't have one

If you wish it to be a non-looped song instead, look into the basic non-looped BGM swap section for how to set the base sound up, then continue on from here.

The popularity of doing this is very staggering, I have to say.

Pick a loaded SCD to start as a template. For this example, you want to choose an in-game song that loops by default (so something like Limsa's day theme - ***music/ffxiv/BGM_Town_Lim_Day.scd***). You could manually set any BGM to loop, but it takes more effort and ain't nobody got time for that.

You can paste these paths directly into the *Loaded SCD* field and press Enter on your keyboard, and they should turn yellow and be recognized immediately. If nothing loads, double check that your path is a valid game path and typed correctly.



Click on *Audio 0* back in the main window once you've selected the BGM, so that we can work with the loop points and replacements.

Your song ideally should be in **OGG** format. It can be **WAV**, but it will auto-convert to OGG when you import it into a BGM-type sound unless you delete *Audio 0* and create a new one. Note that, since late 2024, WAV files have not been behaving consistently when used with loops. Therefore, it's recommended to use OGG for the time being.

If you have an **.mp3/.flac/.m4a** (etc.), convert it to **OGG** or **WAV** with any software or site you choose before you import it.

Now look for the rightmost button next to VFXEditor's built-in player and click it to begin searching for your song.

Navigate to where you have your song saved, select it, and click *Ok*.

Audio 0 will have closed itself to refresh the data; click it open again.

The left number is your loop's starting time, while the right is the loop's ending time. Both are measured in seconds. They are different for every unique OGG/WAV. If you choose to edit these, you may see your inputs change a bit after you type them in. Don't worry - this is normal. Due to how OGG files work with something called pages, the best that can be done is an *approximation of time*. The higher quality your OGG export is, the better this approximation ends up being. That means we'll still have to put in some elbow grease if we want to fine tune our loops, but we know where to start.

If you did nothing special with your OGG/WAV and it doesn't have any relevant *LoopStart* or *LoopEnd* metadata, the left one will be **0.000** because it's starting from the beginning of your song, while the right one should automatically be set to the end of your song. If the right value is also **0.000**, type in an absurdly long value (like 9999999999) and it'll auto-adjust to be the correct ending time. If the file was **WAV**, it may not automatically add the end time, and it won't auto-adjust if you type one in, so you'll have to look for it (in Audacity or the mini player in VFXEditor itself) and then add it in. Assuming you only want to loop your song from start to finish, you don't have to do anything else here.

If you skipped past the section on working with Audacity and want to make custom loop points, you'll probably want to check against the built-in player for some values to start with. Enable the **Simulate Loop Start/End** setting in the *Settings* dropdown to make this a little easier.

> Click around to find some acceptable spots to start and end your loop; type those into the **Loop Time** fields - left for start, right for end. When you put the numbers in, the player will draw green and red lines to mark those points so you can see them better.

If you followed the Audacity preparation at the start of this guide, it's likely you'll want to fine tune the numbers that were filled out for you if your file is **OGG**. (If it's **WAV**, you probably don't have to worry, but it'd be a good idea to double check anyway in testing.)

Now, what a lot of people like to change is what volume channel the music will play on - such as on an orchestrion channel, BGM channel, system sounds, etc. Most have kept it on BGM, but I may be bold enough to recommend placing it on the performance track instead so it can be separately adjusted in volume without interfering with channels that are more frequently used.

> Note: You may need BRD performance unlocked first or may have to swap to BRD once per game session in order to hear your custom sound set to that particular bus of 17. I didn't have that problem, but someone else did, so consider it a game limitation or whatever's happening with their spaghetti code.

Those channels are determined by **Bus Number** and **Priority** in the *Sounds* tab.

Here's a quick reference of what each bus number relates to:

General audio bus numbers:

(undefined): **0**
music: **1**
skills, emotes, cutscene audio: **2**
monster sounds: **2 / 3**
voices: **3**
SE_UI, minigame sounds: **4**
objects (housing, combat, instances): **5**
ambient/background: **5 / 6 / 7 / 11**
footsteps: **6**
limit breaks: **6 / 7**
*placednpc_*: **7**
weird niche BGM cases (*TimeStretchBGM*): **8**
system sounds: **9**
SE_VFX_common: **10**
walla/crowd sounds (*Gaya_*): **12**
title movie tracks: **13**
[unknown]: **14**
[unknown]: **15**
orchestrion: **16**
bard performance: **17**
vibration sounds: **18**
[unknown] (marked as *Housing* in ClientStructs): **19**
*randomwind_*: **20**

You can always check these by opening up those kinds of sounds in the editor.

Additionally, a lot of modders have started using the **Bus Ducking** flag to avoid overlapping music. You can enable this by checking the **Bus Ducking** box under ***Sounds > Sound # > Parameters***. It will create a new tab in this same window, with the following fields:

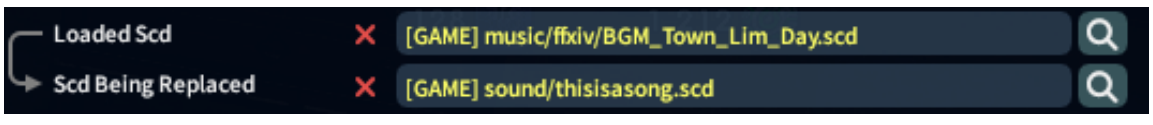
In orchestrions (prior to Shadowbringers), this is set with **Number** 1, **Fade Time** 1200, and **Volume** 0.000. What this does is mutes (**Volume** 0.000) the BGM channel (**Number** 1) for the set **Fade Time** in milliseconds.

You could equally fade or mute any other channel with the right settings. For example: **Number** 17 with **Fade Time** 2000 at **Volume** 0.300 would reduce the BRD performance channel to 0.300 volume over a period of two seconds and keep it there until the sound's finished.

Don't worry about the *Bus Override* fields for this example; they have their uses, but aren't necessary for what we're doing. Just leave them all unchecked and at **0**. If you wish to know more about it, see the corresponding section in *Sounds > Parameters*.

Part of making dance/emote mods also means you'll be adopting a custom path for your SCD. You'll essentially be borrowing the original sound, so you can make all the changes you want in this custom SCD and the original SCD won't be affected.

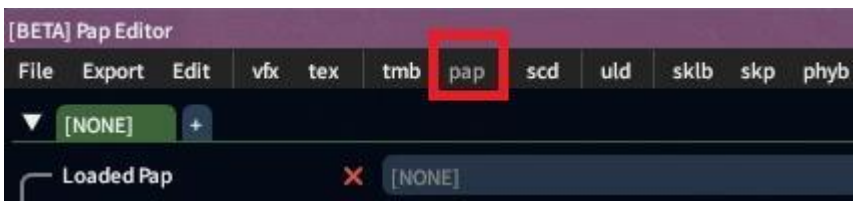
In the *Scd Being Replaced* field at the top of the SCD Editor window, type a custom path - for example, **sound/thisisasong.scd -** and then press Enter on your keyboard. It should turn yellow to let you know it's been accepted. For simplicity's sake, as long as your path starts with **sound/** and ends with **.scd**, it'll work as a custom path.



For further details and clarification on custom paths, and what is and isn't allowable, refer to Appendix F.

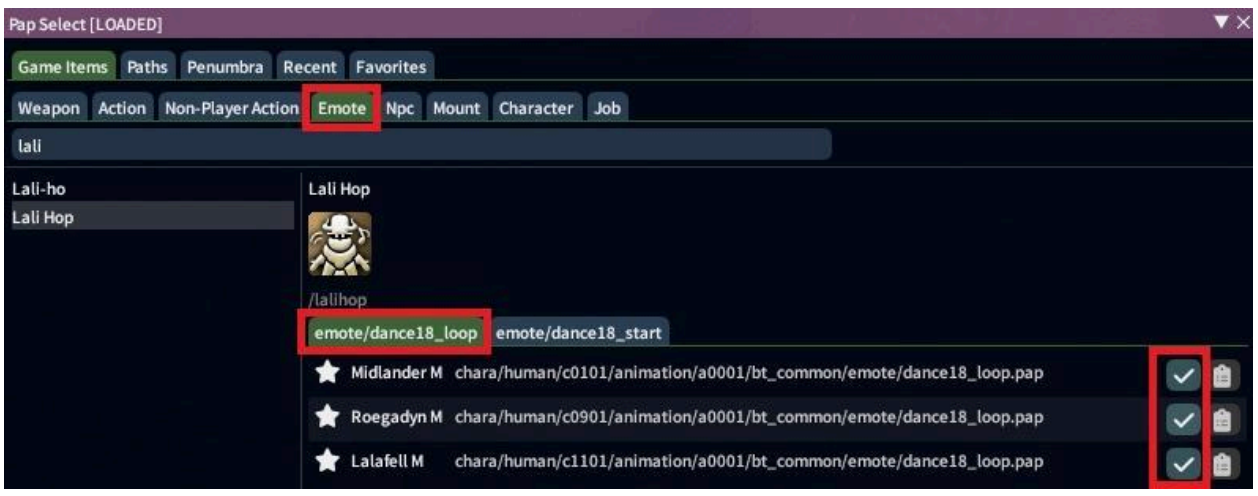
Once you've got all that how you want, click UPDATE to set your sound.

Now it's time to move to another part of VFXEditor that deals with the emote itself. You'll often get better results if you edit the **.pap** file - the one that governs things tied to animation - instead of **.tmb**, at least in the case of dance mods, so we'll go with that.



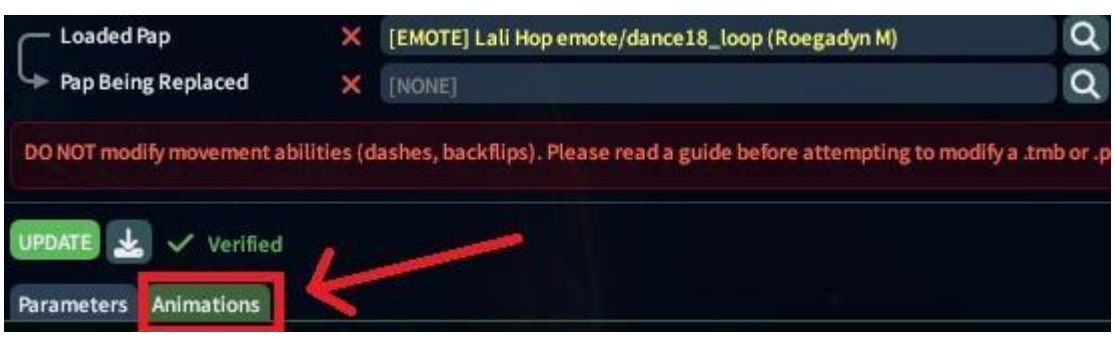
Go ahead and navigate to the emote's PAP you want to use, in the same manner as you did with finding an SCD. The catch, however, is that you have several to choose from because, very often, emotes will differ for certain races (or even all of them). That does mean you'll have to edit all of them the same way if you want to release this for everyone, but we can worry about that after all's said and done. In general, also, you just want the files in the tab that's marked with ***_loop***, unless you specifically want to have a separate intro song for your emote.

For our example, I'm going with **Lali Hop**:



Ignore the movement abilities warning back in the main window, as this doesn't apply to any known emotes. It's strictly for the TMBs of forced movement skills like gap closers.

Open the tabs up in your selected emote as follows:



Add a new track under this actor.

> The more you know: Adding a new track isn't actually required. You can work with any existing track and have as many entries as you want in each one, but this helps to keep everything more organised.

Click **+ New** to add an entry to the track. In the menu that pops up, you want to click the one that says **C063 Sound**.

You should now have this when you open up the entry:

First things first, let's add in our custom path that we established in the SCD editor. Delete *sound/replace_me.scd* in the *Path* field and type in the custom path (or copy and paste it from earlier).

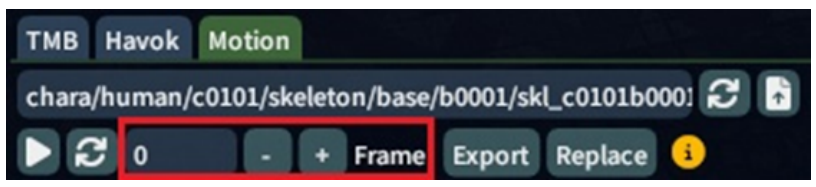
If you've glanced at other guides before this, they may have mentioned that the **Loop/Duration** field (previously *Unknown 1*) should be set to the length of your song in **frames** (30 frames per second). This is also a viable option, and can be used to shorten your sound if necessary, but **Loop/Duration** at **-1** - meaning an infinite duration - will default to using the SCD's loop length automatically without needing to calculate anything.

So, for the sake of simplicity here, change the value of **Loop/Duration to -1** and then check **Stop On Animation End**. These settings will enable the music to continue/loop for as long as the animation plays, while also allowing the music to stop when the emote stops. You can also check **Use Min Max Range** if you'd like; it's technically optional, but this will add a fade to your sound when you zoom out or move away from the source, assuming the SCD has workable ranges.

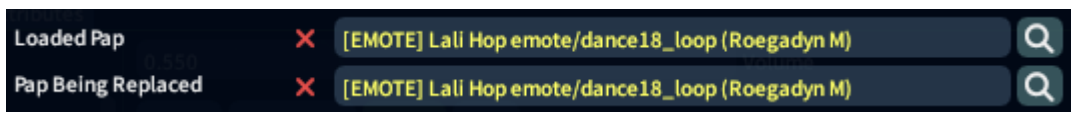
If referencing an older guide, those two checkboxes enabled have a *Sound Position* of **3**, while all three boxes checked have a *Sound Position* of **7**. (Values: 1 + 2 + 4.)

> Side note: If your song path changes a bit in upper/lowercase (like going from *sound/MySongName24.scd* to *sound/mysongname24.scd*), this is intentional. It's to make it compatible in programs like TexTools that are case sensitive. Your sound will still play like normal.
>
> The more you know: How **Use Min Max Range** works is defined in the **Min Range**, **Max Range**, and **Range Volume** values of the SCD itself, found in ***Sounds > Sound # > Layout***. Additionally, if you check **Use Bind Id**, you can set a manual **Bind Id** based on the EID file of your character's race/gender combo. This is usually **71**, or right between your character's feet. If unchecked, it'll default to **0**, which is more or less the same.

You can also change the *Time* field if you want your song to have a delayed start. Be aware this is measured in **animation frames** and **not seconds**. There are **30 frames per second**. If you want to align it with precision, you can go to the *Motion* tab of this editor and play the animation there to see what frame you need to use. The **-** and **+** will move the skeleton in the player below, frame by frame.



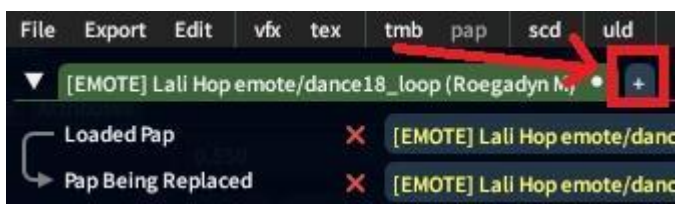
After you're satisfied, set the replacement PAP. For most cases, this will just be the same one that you picked in *Loaded Pap*.



If you wanted to put it on a **different looped emote** (it needs to loop as well in order to work), we have to go a step further and make the animation codes match. Otherwise, the game won't recognise the emote and refuse to play it.

If you don't need to do this, feel free to skip ahead.

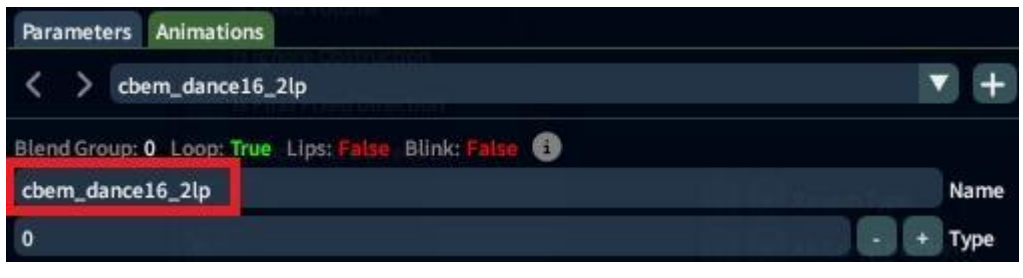
Let's open up a new tab so we don't overwrite our current emote:



In this new tab, load the PAP of the emote you want to change it to. I'll go with **Bee's Knees (Roegadyn M)** as an example.

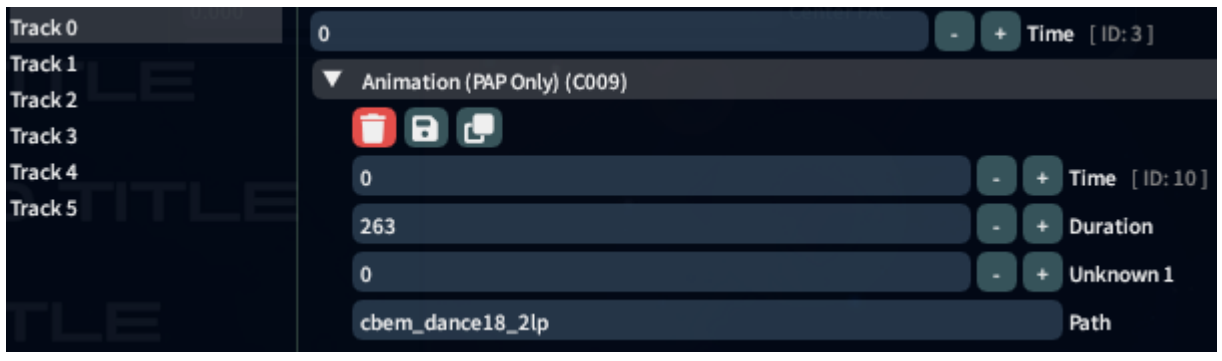
> Side note: You generally want to use the same race and gender combination for animation replacements. If you don't, there's a good chance that your character will deform because it's trying to adjust the skeleton size to match. The exception is if the **base animation is Midlander M**. All skeletons share this base skeleton and will have no problems adjusting to it. In other words, you can swap something like **Lali Hop (Midlander M)** to **Bee's Knees (Roegadyn M)** and it'll be fine, but putting **Lali Hop (Lalafell M)** on **Bee's Knees (Roegadyn M)** will probably make that Roegadyn or Hrothgar look like Mario.

Open the tabs in this PAP to ***Animations > (animation name)***. It's this animation name that we need, so go ahead and copy it. Feel free to close this new tab once you have it.



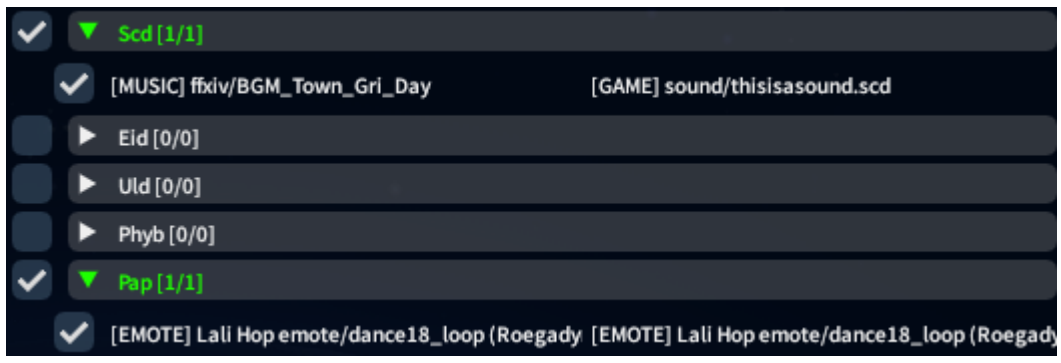
Head back to the emote you were editing. Paste the animation name over the one in here in the same place, under ***Animations > (animation name)***. For our **Lali Hop** example, this will change **cbem_dance18_2lp** into **cbem_dance16_2lp**. You'll see the dropdown menu name change to it as well.

Now in ***TMB > Actors > Actor 0***, find the track that contains **Animation (PAP Only) (C009)** and open it up.



See the field called **Path**? Do the same thing as you did above and replace the animation name. For our example, again, it'll be changed from **cbem_dance18_2lp** to **cbem_dance16_2lp**.

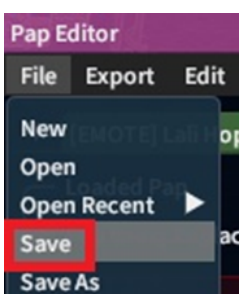
And that should be all you need to do here. Repeat these steps for other race and gender combinations, if you want, then customise and export the complete mod using ***Export > Penumbra***, making sure to select all the files that you need for it to work.



Or if you want to check if it works beforehand, press **UPDATE** to enact the change, which will refresh your character and allow testing in-game.

For fixing your song's loop, volume, or the sound possibly being cut short, look through the basic looped BGM swap testing section, but you can safely ignore the warnings about being in the same zone when you test. As you have a custom path for your sound, that won't matter. If you run into other issues, please reference the troubleshooting section of the guide.

At this point, you may consider saving all your work into a workspace, in case you need to edit it again later. This is simply done with ***File > Save***.



More importantly, though, it'd be a good idea to reset all your changes back to the vanilla versions. You can do this with ***File > New***, then changing zones or redrawing your character.

Adding a sound to a skill, animation, or VFX

This method will cover adding a **new**, **non-looping** sound to any given skill, animation, or VFX. (For looping, please see the previous methods.) I will assume you've either set up a custom SCD to use or have a sound path ready. If you haven't made a custom SCD - in other words, if you want to use a purely custom sound - please check the section on basic non-looped BGM swapping and then Appendix F on how to do so.

Method 1: TMB / PAP

This is arguably the easiest way to add, or even remove, a sound. However, this comes with a few caveats:

- A few TMB are used across multiple skills. This is especially true for some more recent AoE skills, where they have *chara/action/normal_hit/normal_hit.tmb* instead of one that's unique. Take care that you're not replacing a generic TMB.

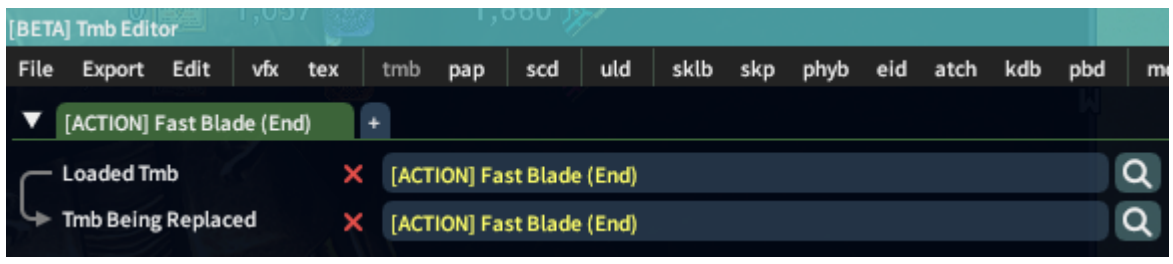
- Some TMB need special attention so as to not disturb some server-side data. This is only a concern if the TMB has a yellow or red entry in it. (Examples: RDM's leap and backflip; skills that briefly lock your movement; enemy skills where they become untargetable.) If it does, **you can still edit it**, but please consult the TMB/PAP document or contact me directly so you can stay safe while doing so.

- There is a very specific PAP that's shared by **a lot** of skills across almost all jobs - this is *action.pap*. In the event you're editing an animation that has this, you will either want to edit the TMB instead or try the VFX method. You can also refer to this guide by Alchemic Potato/Excesum for circumventing *action.pap*. The reason for this is mainly that quite a few mods already edit *action.pap* (like teleport mods or some large-scale job reworks) and you can only have a single *action.pap* active across all mods; and *action.pap* will have to be updated every time a new job is added to the game, or else the new job's animations will be broken while trying to reference your old *action.pap* that didn't have those.

- For PAP that loop (like channelled skills or infinitely long emotes), and with a sound that *doesn't* loop, that sound will play each time the animation loops back to the start. VFX will act the same way. Thus, you may want to place it on the TMB instead, as that will only be called when you first use it.

First things first, load up your TMB/PAP in the Tmb Editor/Pap Editor. For our example, we'll go with Fast Blade on the TMB side. While we're at it, set your replaced TMB/PAP as well. Generally, you want to find the *End* TMB/PAP. *Start* only matters for casting loops, while *Hit* only matters for AoE skills.

If working with TMB, it's heavily advised to keep it the same; this is because of the third caveat above - replacing with a different TMB can get you in trouble if you don't double check its contents to make sure it won't affect server data in some way.

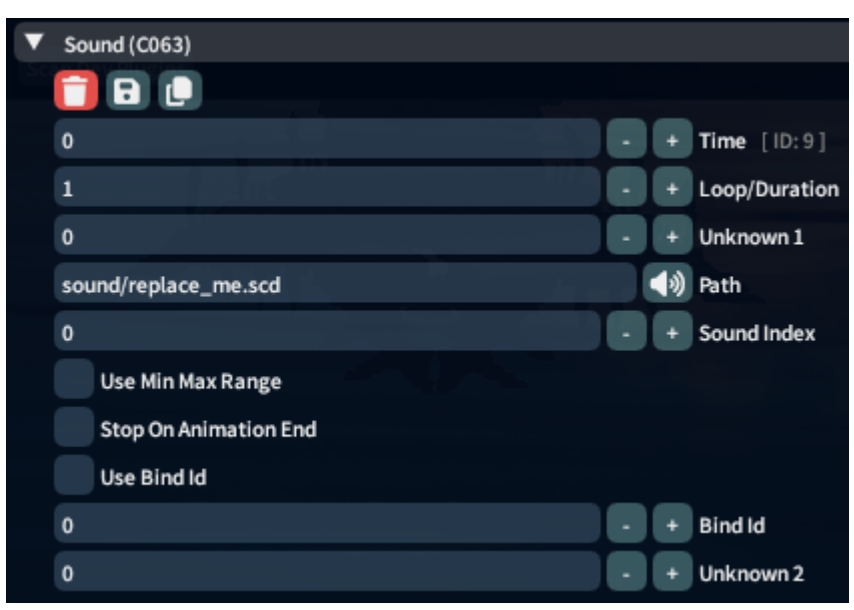


In the *Actors* tab, go to **Actor 0** if you want the sound to apply to yourself or **Actor 1** for the sound to apply to a target. PAP never has Actor 1, since animations apply only to the character that used it. Also, some TMBs won't have Actor 1, namely skills that don't require a target. In those cases, a unique Hit TMB may exist that has an Actor 0, which in that case will apply to all characters hit by that particular skill - an example would be Holy or Quick Nock.

With the following image as a guide, (**1**) go to your desired actor and (**2**) click the **+** to add a new track (**3**). (**4**) Click **+ New** in that new track to open up a dropdown menu. (**5**) Click on **C063 (Sound)** to add a new sound to that track. Repeat steps **4** and **5** for each additional sound.

> The more you know: You can also use an existing track to add your sound. Additionally, there is no limit to how many or what kind of items are in a single track, although it's recommended to keep functionally similar items together as a way to keep things orderly.

Click open the new **C063** entry to reveal this:



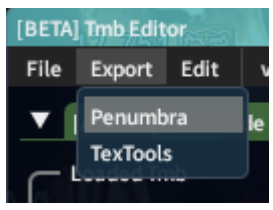
Paste your path in the **Path** field (replacing the *sound/replace_me.scd*) and press Enter. Your path will become all lowercase if it wasn't already; this is intended for compatibility and will not change how it works. Modify other fields as you wish - check the TMB/PAP document for what these do. Repeat for any additional sounds. An example of a modified C063 entry:

At this point, if you want to specify if something should be user-only or everyone-but-user, you'll want to add a Lua condition. As it's fairly involved, please see Appendix G for an explanation, but we'll keep going here.

Click the green UPDATE button to update your changes and refresh your character. For TMB, click the *Play* button to preview the TMB. Be sure you're on the job it affects or you'll run into playback issues. For PAP, the only way to preview is to manually use the animation you've modified.

Repeat for any additional TMBs/PAPs. Click the **+** button at the top of the Tmb Editor/Pap Editor window to add a new tab and keep your changes in the previous TMB/PAP.

Once you're satisfied, click *Export* at the top of the editor to turn your edits into a mod. Note that if you don't export your changes in some way, it'll all revert back to the vanilla game version on restarting the game or the plugin, so be sure to do that.



Method 2: VFX

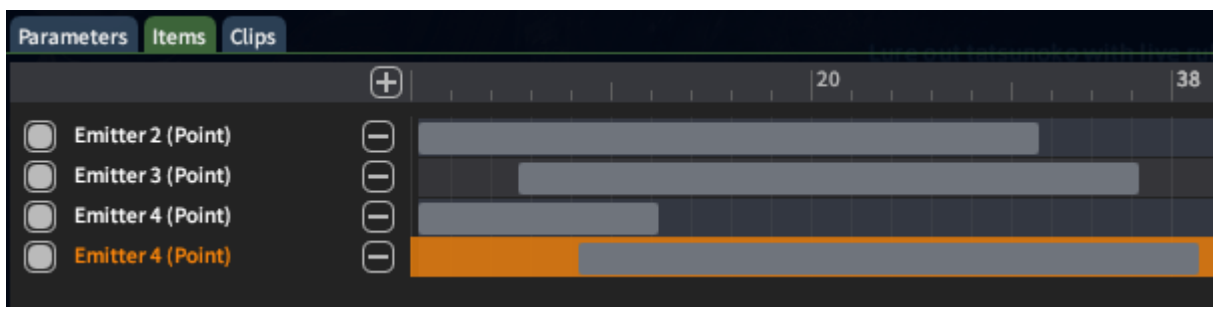
Be aware that attaching sounds in this manner can cause them to be cut short or not play in certain situations. As of patch 7.3, sound handling changed slightly, so you may also find that certain third-party sharing plugins don't behave with looped SCDs attached to VFX - try through PAP/TMB instead if you're having issues with it.

This method gets a little bit more in-depth with the editor. While it can get slightly technical, I'll try to simplify it for the purposes of this guide.

Find the VFX that you wish to attach the sound to. You can do so in the *Loaded* pop-up in the VFX Editor window. Pressing the ▸button should display it on your character. If it doesn't, it's lacking a binder (and that's fine - we'll get to that). You can also preview it with the *Spawn* button if there's something in both the loaded and replaced fields, and this will also give the option to spawn on a target or on the ground.

In case you haven't already, set your desired VFX to loaded and replaced. For our example, we're sticking with Fast Blade #1:

Our goal is to add the sound on an emitter, but the question is which emitter? Well, it depends. For starters, you might want to check the *Timelines* tab. Find *Timeline #* > *Items*. There can occasionally be multiple timelines, but we want to find the ones that have emitter entries and not ones with clips. You'll likely notice that each emitter is offset differently, assuming they're not infinite bars. These are according to what frame they should start. (Remember: 30 frames per second.) You can find out the exact start and end frame if you click on the emitter here and look below it.



> Simple explanation of everything: Emitters call particles - which are basically the meat of VFX, all the shiny stuff - and emitters need binders so that they know where to emit from. Binders are used to bind an emitter to a specific point. Sometimes emitters call emitters, but we're not going to go into that rabbit hole. Timelines will then tell the VFX which emitters and binders to use; they also specify clips, which do things like cancel VFX if a weapon is put away or a buff runs out, or set up randomised timelines. And while we're not concerning ourselves with effectors, those things can provide VFX-based lights, camera shakes, and radial blur.
>
> If you're curious about all the details, luckily for you, I've got a VFX document set up for that very purpose, but you won't need it for this example.

If you think one of these timeline emitters will fit the bill in terms of start time (and possibly length), then you've found the emitter you want to attach the sound to. If not, then we can just make our own with a little extra effort.

For the sake of our example, let's make a new emitter and leave the rest intact. However, if you want to work with an existing emitter, it's basically the same except you'll just be selecting that emitter to work with in the dropdown in the *Emitters* tab.

In the *Emitters* tab, click the **+** button next to the dropdown where you'd choose the emitter to edit. Click **Default** in the pop-up. This will have created a new, empty point emitter in the dropdown, and it will always be the very last one in the list. Go ahead and click it.

Under the *Parameters* tab, you'll notice the following below the node graph:

To add a sound, click that **+ Sound** button and change the **Sound Index** value to **0**.

> Side note: If you happen to be working with an SCD that has *multiple* sound indices inside of it (like *SE_VFX_Common* or *SE_UI*), you'd instead make **Sound Index**'s value, well, the sound index that the audio index uses. Otherwise, single-audio SCDs will always use **0**, and anything higher than **0** will default to 0. Also, **-1** in this particular field means that it's not being used (which the little ***i*** tells you if you hover over it).

In the newly created **Sound** field, paste the path to the sound you want to use. It will automatically convert to lowercase; this is for compatibility reasons and doesn't affect how it plays.

For **Loop Start** and **Loop End**, we are doing nothing with this. Despite the name and its proximity, it's only for looping the *emitter* and not the sound itself. If you wish to establish a loop for the sound akin to BGM, you want to do that through SCD metadata and/or PAP/TMB C063 manipulation.

If you were using one of the existing emitters, you're done here, and you can test it by clicking the UPDATE button and then the *Spawn* button. If it's what you like, you can move on to exporting your mod or changing additional stuff in the editor.

However, for our example, where we made a custom emitter, now we need to actually tell the VFX to use this emitter. Head back to *Timelines > Timeline # > Items*. Click the **+** button there to add an empty item in that timeline. (You'll see a red **!** pop up above, but it'll resolve itself once we input some values, so don't worry about it.) Click that new item and look below.

We want to change its **Target Binder**, **Target Emitter**, and double-click the **Enabled** checkbox so that it turns into a checkmark. The *Name* only saves if you have everything set up as an editor workspace, so you can ignore it.

For the binder, there's normally just one and that's what we'd use, *but* if you have multiple, you can check what they do in the *Binders* tab. Mostly, you'll want to concern yourself with the fields in *Binders > Binder # > Properties*. In the Parameters tab of the binder, it'll tell you if it's being placed on caster or target, as well as what bind point ID it should adhere to (defined in the EID, another file type that defines binder locations according to a skeleton). Pick the one that you think would work best. You could make your own binder, too, if you felt like it.

> Side note: VFX for ground AoEs and objects typically don't have a binder, and so you wouldn't need to add one to your emitter either. Even if the editor's complaining about it, it's fine, since that binder is specified for the VFX elsewhere.

Turning our attention back to the timeline - for the emitter, just choose the one we've made.

Lastly, we want to modify the **Start Time** and **End Time** for the timeline entry. In case you've forgotten, 30 frames per second. Make sure that **End Time** lasts as long as you need it to, or else your sound will be cut short - unless that's your intention. You can set this to **-1** if you know your VFX is set up to loop (like if it's attached to a mount or tied to buff timing); otherwise, you'll want to match it with your sound's length.

You can also do the same thing by dragging the bar in the timeline. Drag from the middle to shift its placement, and drag the ends of the bar to adjust its length.

Test by clicking the UPDATE button and then the *Spawn* button. If it's what you like, you can move on to exporting your mod or changing additional stuff in the editor.

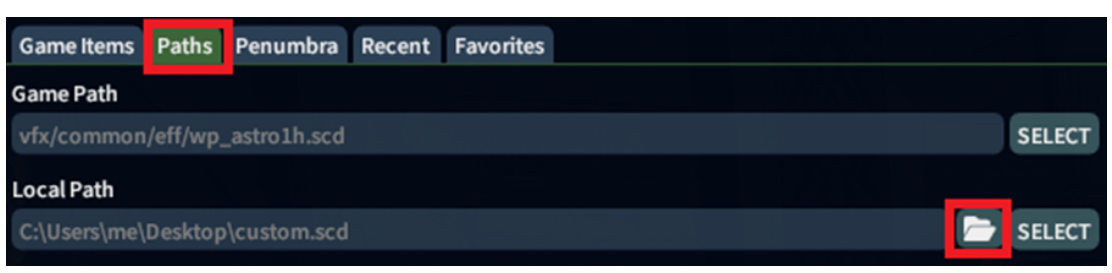
Replacing an existing modded sound with your own

This example assumes you only want to change the sound internally and don't particularly care about doing much else to it. If you feel like going the extra mile and making the SCD file from scratch with some extra settings, you can refer back to the first example about BGM swaps. It's basically the same idea with a slightly different end result.

Load up the sound on the mod you want to edit. Click the button next to the **Loaded SCD** field and navigate to your mod. You can do this by going to the **Paths** tab and typing in the path or finding the local path with the folder button; or going to the **Penumbra** tab and searching that way. Click ✓ (or **SELECT** in the **Paths** tab) once you've found the sound you want.

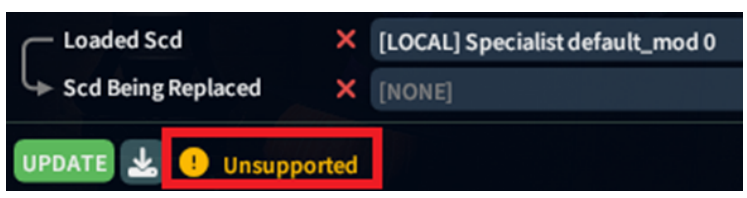
For Penumbra:

For local files:



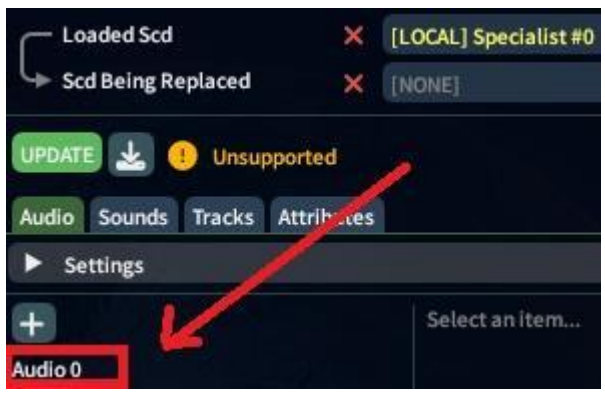
> Side note: Some mods may make use of an intro sound that's not looped and then a normal sound that's looped. These go along with animations that have both a start and loop. If you don't want to keep the start SCD and just replace the loop SCD, you can just delete the start SCD path in the redirects; I'll get to this concept near the end. Or you can manually move/rename/delete the physical file so that the mod doesn't run it. That also works.
>
> If you want to replace both, you'll have to manually split your sound to have both. However, you may run into an issue if the start sound isn't of identical length - the sound can be cut short, overlap, or leave a noticeable gap, due to how the original mod had things set up.
>
> > **Solution one**: Adjust the start PAP (animation) to have a duration to match your new start sound. It will mean the animation altogether will look strange or freeze a bit, but it works.
> >
> > **Solution two**: Make a start sound of equal length to the original start sound, then set up your loop points of the looped sound manually. Not to sound like a broken record at this point, but the first example, the Audacity section, and the SCD prep section cover how to do that, with the latter two links being a little more technical/wordy in nature.

For some imported sounds from mods, you may see this:

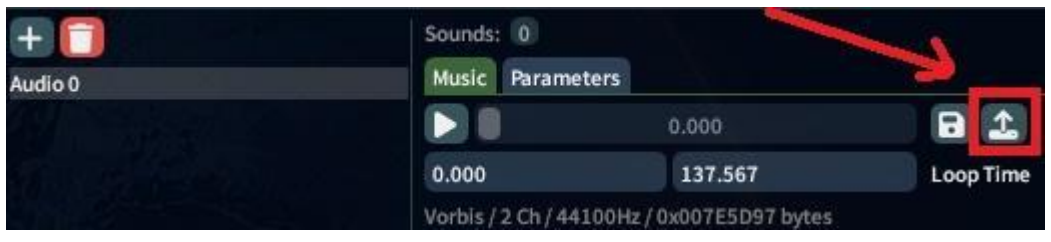


It just means the conversion was done outside of VFXEditor and has the potential to run into issues. It's still entirely functional. However, if you find the sound isn't working how it should be, it's recommended to go back to the first example and make your own SCD.

Click on **Audio 0** to bring up the built-in player.



Then click on the rightmost button next to the player to import your custom sound. The editor will take care of any possible conversion.



> Side note: For music/BGM, it's recommended your file be **OGG** with the highest quality setting. For things like battle sounds/SFX, it's recommended to be **WAV**. You can import whichever type you want, really, but keep in mind the overall size. If the SCD's original sound was **OGG**, VFXEditor will automatically convert **WAV** into **OGG** upon import, unless you delete **Audio #** and make a new default template to force it to be **WAV**. Note that, since late 2024, WAV files have not been behaving consistently when used with loops. Therefore, it's recommended to use OGG for any looped sounds for the time being.
>
> HCA is also allowed, but this encoding should be considered experimental, as it was only added to the game in patch 7.5.

Any other file type besides WAV, OGG, and HCA is not supported, so you need to use an external software or website to convert it.

If you're unsure what type of sound the SCD has, look below the built-in player. If it says **Vorbis**, that means it's **OGG**. If it says **MsAdPcm**, that means it's **WAV**. HCA is **HCA**.

The player will disappear as it loads the new sound. If it's a large file, give it a few seconds. Once it's done, just click on **Audio 0** again to bring it back. For some folks, **Loop Time** may have changed.

This is the editor adjusting the length to match the end of your custom sound. For the most part, you don't need to worry about it unless you want to specifically adjust your loop points. Please refer back to the first example about what to do in that case.

If it hasn't changed and you need the sound to loop back from end to start, type in a very large number into the rightmost field of **Loop Time**. It should automatically set it to the end of your sound. It may not match up exactly, depending on your sound's overall quality, but it will work as intended.

If it hasn't changed and you don't want a looped sound, you don't need to do anything. Both **Loop Time** fields at 0.000 will default to a non-loop status.

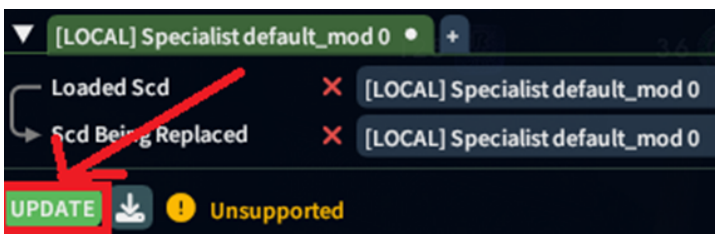
If you're having problems during testing, with it not looping when it should or looping when it shouldn't, double check the following:

- the **Loop** flag in ***Sounds > Sound # > Parameters***. Check it for loops, uncheck to disable loops. This may or may not be necessary.

- the correct sequence of **KeyOn**, **(LoopStart)**, **Interval**, and **KeyOff/LoopEnd** exists in ***Tracks > Track #***. For what you should be looking for, check here.

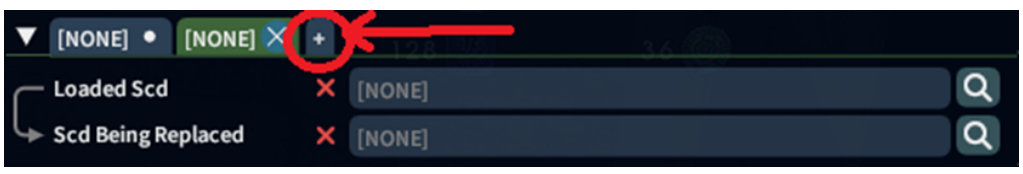
- the file that's calling the sound isn't affecting it. A PAP or AVFX can force looping status sometimes. Assuming this file exists, check for a **-1** in the PAP's **C063** entry or a relevant field in the AVFX. (AVFX files have a lot of influencing factors, which can make finding the culprit difficult on occasion, so if this is the case, you may want to ask for assistance.)

When working with a mod loaded and active in Penumbra, you can test your sound if you wish by navigating to the mod's sound again for the **SCD Being Replaced** field. Click UPDATE once you do that, then use the mod in question to hear it. If you're replacing BGM, make sure the old one isn't playing when you click UPDATE or you risk crashing.



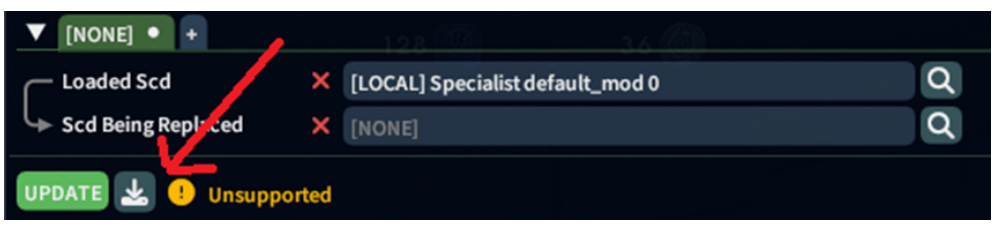
Side note: In case you're curious or worried, the UPDATE button itself won't actually overwrite your mod's sound yet. VFXEditor stores it as a temporary file and references that until you choose to export it.

Additionally, if you plan to work with multiple sounds or files, you need to have them open in multiple tabs. You can add more tabs like so:



Clearing the **Loaded SCD** with **X** in the same tab will also clear your temporary file and revert all your changes.

In order for the sound to actually be more permanent, you need to export it. Click the button to the right of the UPDATE button to do so. Pick somewhere to save the file and give it a name. It can be the same name as the original mod if you just want to overwrite, *or* you can give it a new one if you want to give the mod toggle options. We'll get to how to do that in a second.

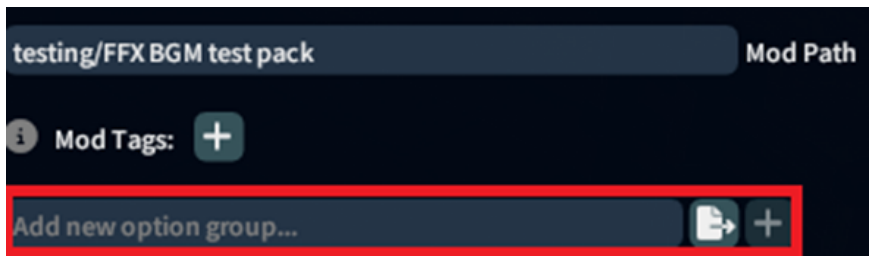


Find your newly created SCD file, wherever you saved it, then copy and paste it into the mod's folder where the old sound is. For doing a straight replacement of the mod's sound, simply change the new sound's name to match the old one (overwriting the old sound or making a renamed backup somewhere) and you're done. Otherwise, if you're making options, keep the name how you had it.

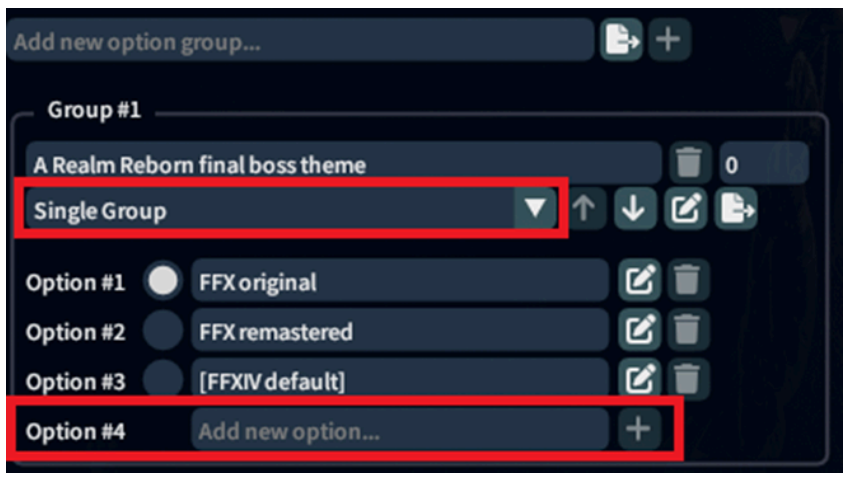
With creating a new mod option, we have just a bit further to go. I'm going to assume you have the mod imported into Penumbra already for this, as that's the way to add extra options. I don't know how to do any of this with Textools, and I don't particularly encourage the use of Textools anyhow because it does bad things to your index files by directly modifying them, but I digress.

Open up Penumbra and find the mod you're editing. Go to the **Edit Mod** tab in that mod.

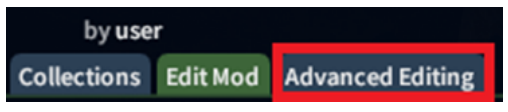
Find the field that says *Add new option group...*, type in anything you want for the new option group name, then press the **+** next to it to add it.



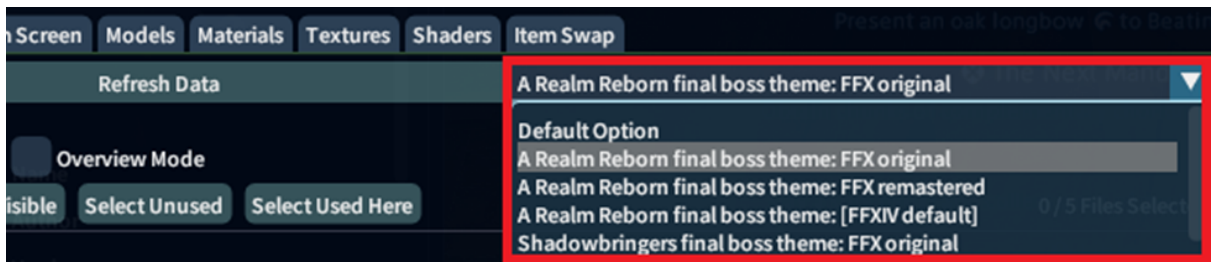
Make sure the group is listed as type **Single Group** so that you don't get conflicting or overwriting sounds in your mod. Basically, it just forces you to only have one of the options active at any one time. Then, type in an option name in the *Add new option...* field, and press **+** to add it. Since we have at least two sounds - the old and the new - make an option for each of them. I also add an extra option in the event I or any other user doesn't want a modded sound at all.



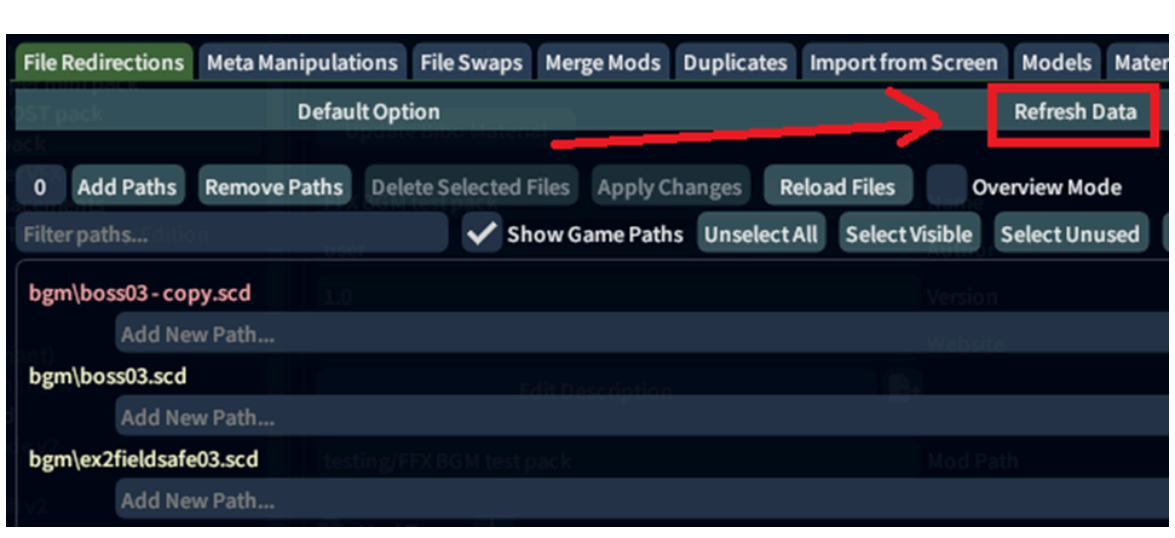
With that done, go to the **Advanced Editing** tab.



To set up a sound for each option, click the dropdown to the right in the **File Redirections** tab (it should be opened to this tab by default) and click the option you want to add the first sound to.



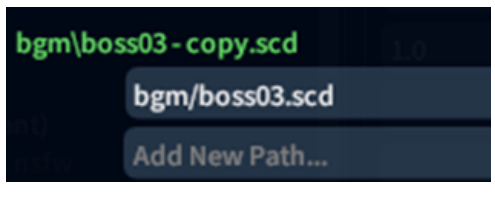
You should see all the sounds that exist in your mod's folder(s). If you don't see the one you just added, click the **Refresh Data** button.



> Side note: Red means you haven't assigned a path to the file yet to make it usable in-game. Yellow means there's at least one option that has a path assigned already. Green means the path is assigned for the option you selected in the dropdown menu.

If your mod didn't have any options established yet, the original sound will have its path already set in the **Default Option** in the dropdown. That just means it'll be loaded regardless of whatever option you pick. It also means you'll need to cut the path from there and paste it into the option you want to avoid conflicts. Note that, if you changed any path redirect in an option, you'll need to click **Apply Changes** at the top before moving on to a different option, or else everything will revert to how it was.

For basic mods, you really only need to use the same path but just in different options.



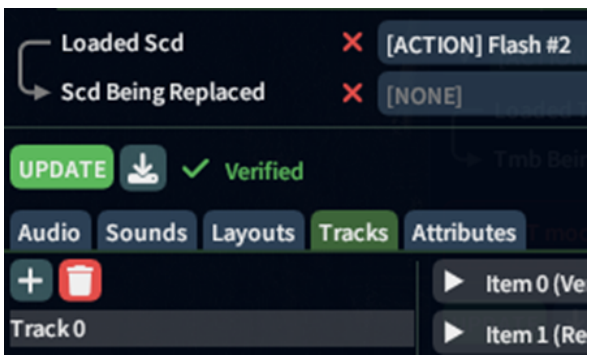
And if you did what I suggested, by making an extra option to disable the modded sound, you don't have to add any paths in that option in the dropdown. It will assume that nothing will be used and will default to either what's in the **Default Option** or what's in the game, if anything.

And there you have it - new sounds and options for your mod.

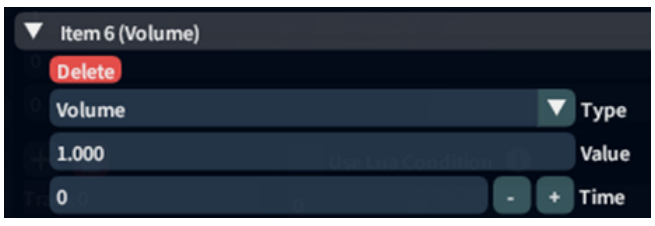
Cropping sounds to only play specific sections

Be aware that this example is a little more technical in nature. If you find it difficult to wrap your head around, you can just use Audacity or something to cut your sound how you like before importing it to replace another. Doing manual cropping in an outside program is preferred for this sort of thing anyway, especially in cases where you want the desired section of sound to play immediately. While you can manipulate the SCD internally to achieve that with some clever pitch additions - I'll cover that concept near the end - it's usually more effort than it's worth. But for those of you that are feeling lazy or daring, we're off to see the wizard.

Let's say we want to change Flash's SFX to only play the second half of it. Go ahead and load the sound (*sound/vfx/magic/SE_VFX_Magic_Flash_c.scd*), set its replacement to be the same, and go to the **Tracks** tab, then click on **Track 0**.



You'll notice quite a few item listings here. If you want to read about what they all do, check the Tracks section of the guide, but for our purposes, we're only interested in working with **Interval** and **Volume**.



Interval you may have read about or tinkered with already. It's how the SCD determines when to start, what modulations to activate at that time (pitch, panning, volume, reverb, etc.), and how long to play parts of the sound. The placement of the **Interval** field is crucial to how it functions, but we'll get to that in a second.

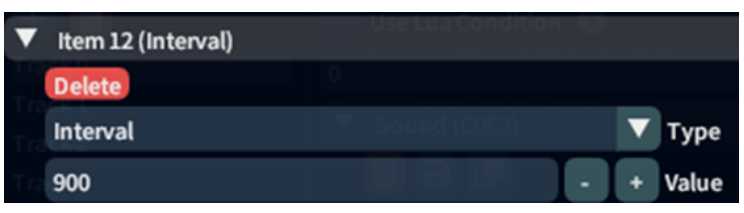
Every entry before **Interval** is what will be used when the sound is called. In Flash's SCD, the ones that currently hold usable values are **RandomPitch** (1.059 to 0.944), **Volume** (1.000), and **Pitch** (1.000). The rest are either set to off with **ModulationOff** or by having a **0** value.

The important part is the **Volume** listing at **Item 6**. Since we don't want to hear the first half of the sound, we set **Volume** to **0.000** (i.e. silent). This won't negate any of the modulations that come before or after it, so don't worry about that.

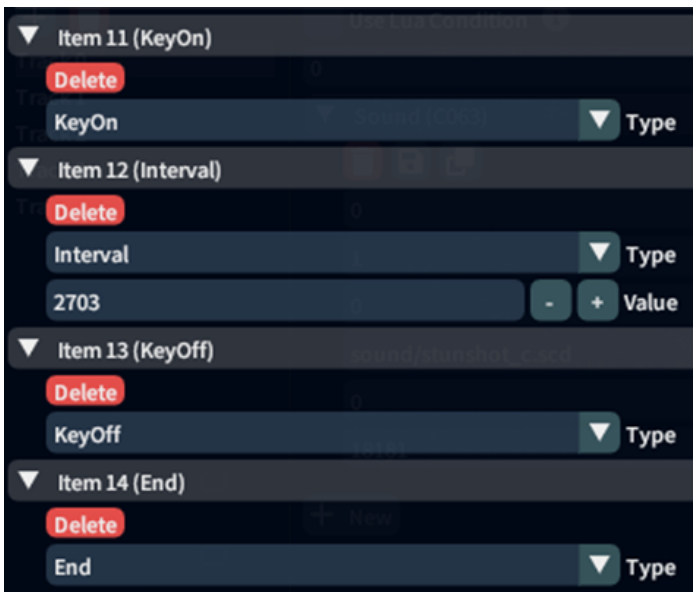
Now, let's hop to the **Interval** at **Item 12**. Right now, it's set to **2703**, the length of the sound in milliseconds. If you preview the sound with the editor's built-in audio player back in the **Audio** tab, you'll notice the elapsed time in seconds in the middle of the bar. This will help you find about when your desired crop will be. Since it's shown in *seconds*, just remove the decimal place and you have your millisecond value.

For our example purpose, let's say we just want the pop sound and everything after. That happens at roughly 900 milliseconds. Therefore, that'll be our **Interval** value.

> Side note: any variation in **Pitch** or **RandomPitch** outside of 1.000 will also affect overall play speed independently of any millisecond values set in the SCD, so keep that in mind.



And then we want to change our volume. But wait, there *is* no **Volume** entry after it!



Okay, see that fancy **+ New** button?

Go ahead and click it. Now we have a new item.

"But it's not the right one!" Patience, young Padawan.

This is where order is important. You always want a **KeyOn** before **Interval**, and you (almost) always want **KeyOff** followed by **End** at the end of your series of items. I say "almost" because you might sometimes see **CutOff** or **LoopEnd** instead, but we won't get into that here. If you don't have those four items, your sound won't play. The exception is with sounds crafted in XIV File Explorer, in which they only have **ReleaseRate**, **KeyOn**, and **End**. They're based on an outdated SCD setup in older SE games and are made to function as such, but that's definitely not ideal for what we're trying to do.

So, in order to rectify our current situation to have both **Volume** and the correct order, we'll want to adjust the **Type** of each item using the dropdowns to eventually end up with this:

At this point, that should be all you have to do. Flash will only play the second half. You can, of course, go even further by alternating more **Interval** and **Volume** items inside **KeyOn** and **KeyOff**, hacking and cutting away at your sound until it doesn't even sound like the original anymore. Or you might slap in some **Pitch** and **ADSR** in between. It's entirely up to you. The world's your special, little oyster.

Since we're here, we might as well get into how to skip sections of sound entirely without having to sit through silence. You've likely intuited how to do it by now, but I'll go over it in case. I'll use the same example sound - Flash.

Basically, the gist is to increase **Pitch** by a large amount and then bring it back to normal. And since it also affects speed, we're abusing that to fast forward through the silent bits.

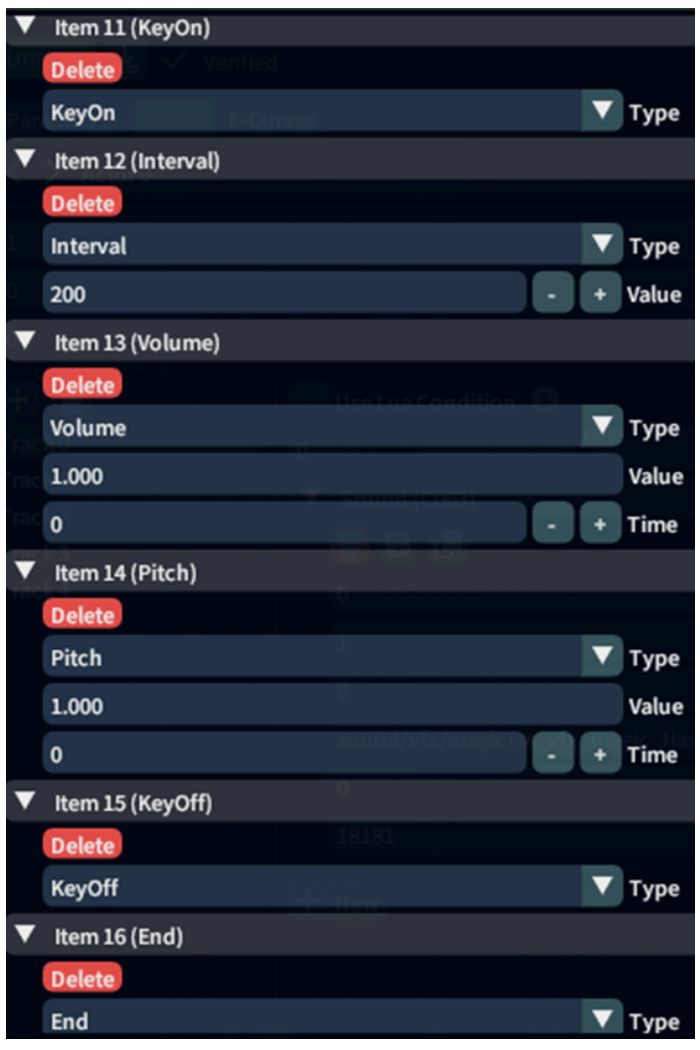
Keep the first **Volume** at 0.000. The very first item we want to change is **Pitch**, **Item 7**. Since this is before an **Interval**, it'll apply right at the start. The problem is ascertaining what value this should be. I'm sure you could do a calculation if you really wanted to, but I like the scuffed approach of trying until I get it right.

However, I'll save you some time and say a **Pitch** of 4.500 will do just fine.

Then we want to adjust the first **Interval** at **Item 12**. Instead of the 900 we decided on earlier, we'll make this 200 to account for the speed change.

This is where we want to add another new item, **Pitch**, and change the last few to maintain working order. We can leave the second **Volume** of 1.000 at Item 13. As long as it's after the **Interval** (and before another one, if you choose to add more), it's fine where it is.

We want our speed back to normal, so that means this new **Pitch** will be 1.000. If you did it right, it'll look like this:



And now if you try it out, your cropped sound should play almost immediately. Or, more accurately, after 200 milliseconds, but close enough for our cheap example. Adjust values as necessary to achieve your desired result.

Enabling the use of randomised BGM/SFX for animations/skills

Note that this can *only be done for animations* (and maybe VFX or non-BGM sounds; I haven't tested that far). I don't know that there's a way to randomise *actual* BGM. Attempting to replace one and force random will mute the BGM altogether, so until I or someone else finds a way to get around that, I'm calling it a game limitation.

While this method works by editing the SCD itself, you can also choose to do randomisation through a VFX with its emitters using separate SCDs. There's a video guide for that process here.

For the SCD trick, we need to borrow a sound that already has a **Type** of **GroupRandom** set up in ***Sounds > Sound # > Parameters***, as well as one that has premade audio tracks. Trying to add a new audio track and using that will end up with that track being silent; this is an editor bug with no current ETA for a fix.

As an example, we'll be using **bgcommon/sound/fst/fst_spot_drop.scd** as our base sound, since it has ten available audio tracks. Paste the path into **Loaded SCD** and press Enter. And since we want to use this in an animation, we'll go ahead and enter our custom path. Type what you want it to be in **SCD Being Replaced** - like **sound/test.scd** - and press Enter.

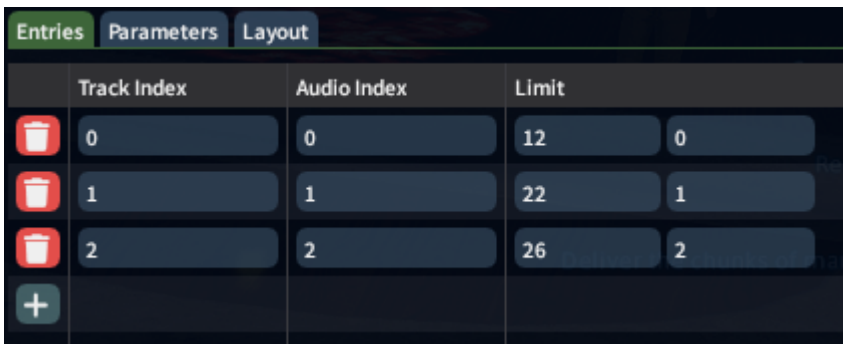
For a quick explanation of what **GroupRandom** does, it defines a series of sounds in ***Sounds > Sound # > Entries*** and gives each one a **Limit**, which is a somewhat fancy term that means it'll assign a **random chance of playing that particular audio in that set**.

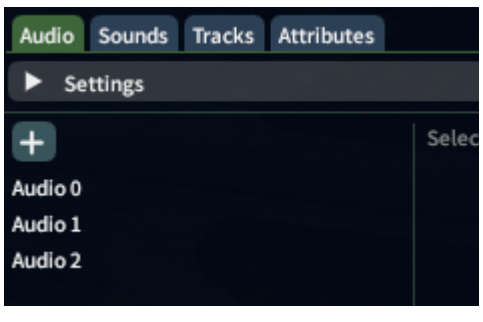
The **first column in Limit** is our **% chance**. The **second column** you might think of as the **order**.

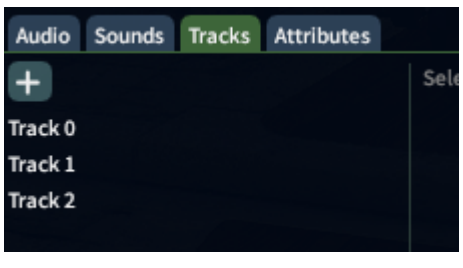
> Testing note: As this *Limit* field is particularly new (at least in its current form), I haven't really dug into it in earnest, so I don't have a ready explanation as to why it can skip around, just that it can. In any case, we don't have to worry about that bit, just that we should stay in incremental order.

As you might have surmised, **Track Index** is tied to which track number in the *Tracks* tab that the *Audio Index* uses, while **Audio Index** is tied to which audio is used in the *Audio* tab. These don't necessarily have to be in order, and quite often they aren't, but it's a good idea to keep them in order anyway to avoid general confusion.

Now, unless you're actually planning to use *all ten sounds*, we'd probably want to trim all of this down. For our example, let's go ahead and do that, and only keep **three sounds**. That means we'll delete everything after **Track Index 2** in ***Sounds > Sound 0 > Entries*** and delete all the audio after **Audio 2** in the ***Audio*** tab. While we're at it, we can get rid of everything after **Track 2** as well in the ***Tracks*** tab; this one isn't required, but it'll be cleaner.



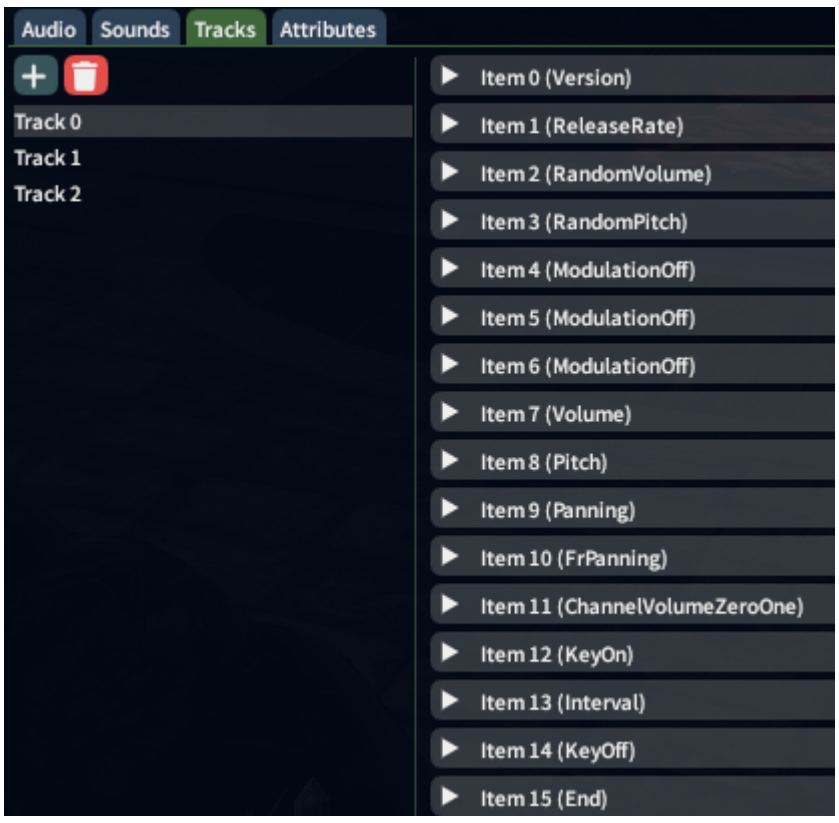




With that out of the way, go ahead and import the sounds you want in whatever format you'd like (WAV or OGG; HCA is currently not supported for import into WAV). Since we're working with base audio that's already MsAdPcm WAV, we don't have to worry about WAV being auto-converted to OGG, if applicable.

For each audio, set up loops (or no loops, whichever) as you would normally.

Then let's take a little trip to the ***Tracks*** tab. For now, check what's in ***Track 0***, which is tied to ***Audio 0***.

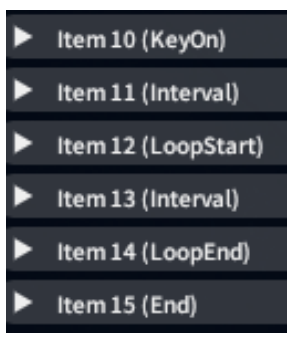


If you're already familiar with this tab, then you'll have noticed a couple things that we need to fix for our needs. First and foremost, we want to get rid of **two items**: **RandomVolume** and **RandomPitch**. As their names suggest, they'll apply randomised volumes and pitches to your sounds when they're played. Unless you really want that, it's best to delete them. Click the items open and press Delete in them.

Our second goal is dependent on your needs.

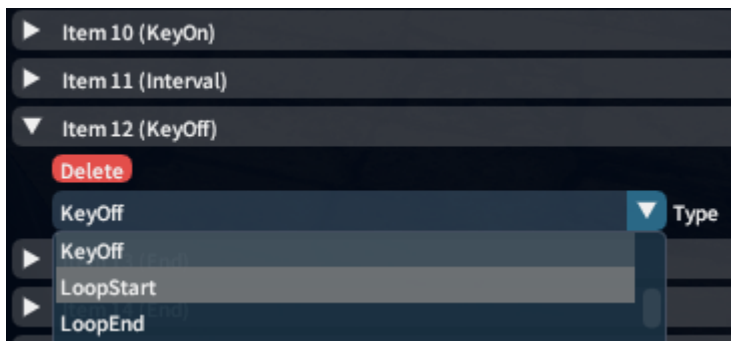
If you don't want your sounds to loop, we don't need to do much here, since it's already set up how we want. The only thing I would suggest is changing the millisecond value inside **Interval** to match your sound's length so that it doesn't get cut short.

If you want your sounds to loop, we need to add a couple items - specifically, we're adding **two new items**, so click the **+ New** button twice to add them. Each new item will default to **End**, but we can easily change that. To give you a quick idea of what we want our items to look like after we're done, refer to this image:



In some cases, the item numbers won't match up. That's perfectly fine. What matters is the order of them.

Going by this image, we want to change **Item 12, 13, and 14**. To do this, click open the item you want to change, then open the **Type** dropdown menu, scroll in that menu until you find the *Type* you need, and click it to set it. Repeat until you have the items in the right order.



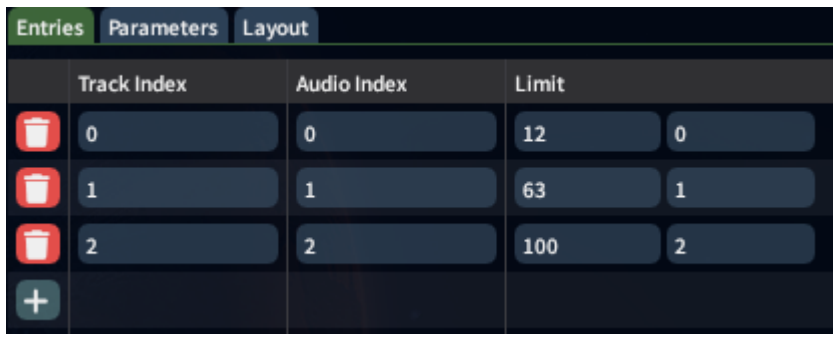
> The last thing we want to do is modify both *Interval* items. To refresh, the **first Interval** is the **start time of your audio's loop**, while the **second Interval** is the **length of the audio loop**. Both are measured in milliseconds.

You'll have to repeat this process for each ***Track #*** you have. Thankfully, each track has exactly the same format, so it's just a matter of doing the exact same thing twice, but for some files you'll have to pay special attention to other things being silently modified - like if **Pitch** has been adjusted from 1.000 or there's some **Panning** going on outside of its default of 0.000.

> Side note: you *can* just have one *Track 0* and set all your audio indices to draw from that. The problem is that you're likely to have each audio track being different lengths. **Interval** usually won't play nicely with some of them as a result.

Now let's head to ***Sounds > Sound 0***. Since we deleted all but three entries here, we need to modify the *Limit* values to behave like we want them to. To further explain on *Limit*, it'd help to think of the first column also as "if % chance fails, go to next entry". In other words, in our example, the first audio we have is **12% chance**. **If that 12% fails, it tries the 22% chance. If that 22% chance fails, it tries the 26% chance.**

You're welcome to make these any percentages between 1 and 100. There's only one requirement: ***Limits* should be unique percentages**. Your last *Limit* should ideally be 100 so that something will always play regardless if the others fail, but you can choose to have it be less if you want there to be a chance of nothing playing. Anyway, something like this would be fine:



> The more you know: SE has *somewhat* of a convention in terms of setting these. They like to space it out to be even amounts between percentages, so something like **32**, **63**, and **100**; or **22**, **43**, **67**, **85**, and **100** - roughly average spacing to give them equal chances.

We're almost done. Hang in there.

Under ***Sounds > Sound 0 > Parameters***, you'll most likely want to adjust the **Loop** checkbox accordingly, change the **Bus Number** (if **5** for the ambient channel isn't your jam), and maybe add **Extra Desc** or **Bus Ducking**.

Additionally, in ***Sounds > Sound 0 > Layout***, changing **Max Range** and **Min Range** to match your target sound type wouldn't go amiss. (For BGM, the default is 0.000 Max Range and 30.000 Min Range.)

If you're unfamiliar with some of the above concepts, you may consider reading this introductory section on relevant metadata.

The very last thing I want to point out is that some of the stuff in the ***Attributes*** tab is a little different than usual, mostly that *Attribute Id, First Condition,* and *Argument Count* have values (and subsequently *Extend 1* has values because of *Argument Count*). For general use, **I wouldn't really pay this too much mind**. Your sound will still act mostly like it should. The only possible difference is that it can define how many instances of this particular sound can exist at one time. At least, as far as I know. *Attributes* is still a pretty big mystery. If you find your sound is getting cut out randomly when other sounds are being played at the same time, this may be something to adjust. As for what to adjust it *to*, just find an SCD that matches what you're going for and copy what's there.

Okay, that should more or less do it for setting up your SCD to randomise sounds. The only thing left to do is to attach it to something. If you're unsure how to do that in general, please see the fourth example in this section for slapping a sound on an emote.

As a gentle reminder, this *does not work for standard BGM replacements. This is a game limitation*.

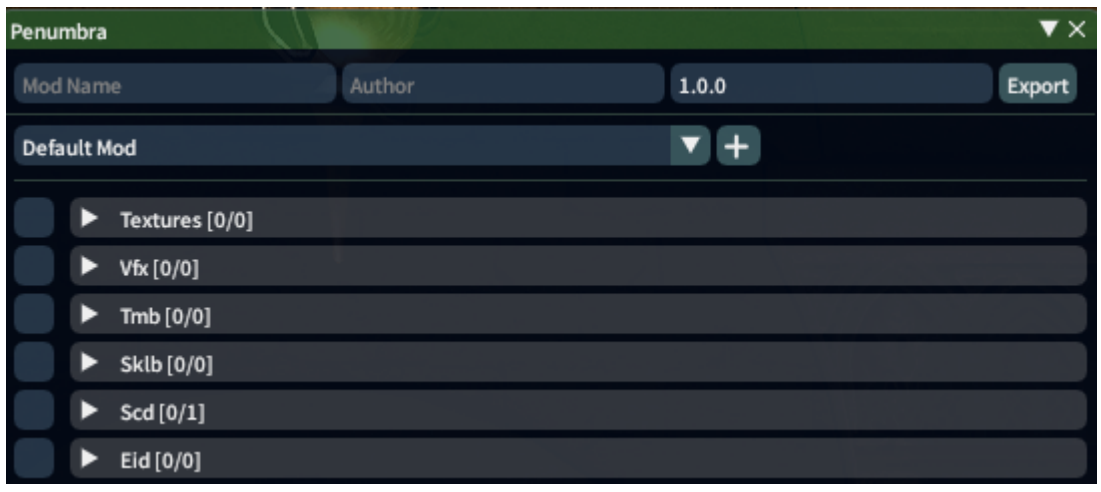
I've got my music and/or sound(s) ready! Now what?

This will be the basics of basics. If you already know how to do file swaps and mod creation with selectable options in Penumbra, there won't be much for you here.

(I've opted to remove the mini section on TexTools mod creation. Use of TexTools is overall not recommended due to how it interacts with the game's index files and its incompatibility with plugins like Penumbra that largely prefer clean indices. Creating .ttmp2 through VFXEditor is still possible with **Export > Textools**, but giving it options requires direct interface with TexTools or making each sound a separate TT file, and I'm not invested enough there to do a write-up. I apologise for the inconvenience.)

If you're exporting as a Penumbra mod from VFXEditor, this method will need a little extra work if you want to add things like mod/group/option descriptions, mod tags, or if you have custom paths for non-emote BGM. Those have to be manually modified in Penumbra itself. However, this is a good starting point, rather than making the mod entirely from scratch.

Click ***Export > Penumbra*** in the editor if you haven't already. A new mod creation window will pop up:

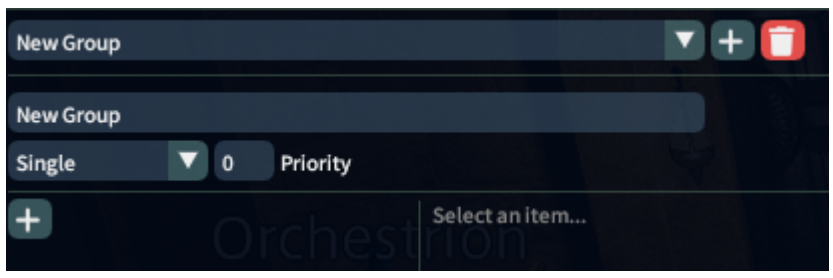


Here, you can give the mod a name, an author, and change its version from **1.0.0**, if you want.

To select all files of a given type for exporting, click the checkbox next to the type. To select specific files of that type, click that type's bar to open it up and show individual checkboxes for each file you have available for export.

You'll notice it says **Default Mod** near the top. This corresponds to Penumbra's *Default Mod* setting when you enable any given mod - in other words, it's what will be active as soon as you click to enable the mod in Penumbra. If you don't care about options, this is where you'll select everything you need.

If you *do* want options, click the **+** next to the **Default Mod** dropdown menu to add a new group (or multiple groups). To swap between created groups, click that dropdown menu.



Below that, you can change the group name from its default name of **New Group**.

The bottommost **+** will add options for the selected group. Click on **New Option** to show everything you can do for that option: changing its name, making it active by default (with the **Default** checkbox), adjusting that option's **Priority** in relation to the other options, and selecting the files you want to add to that option.

The dropdown below the group name tells the mod how the options should work. **Single** will make it so that you can only have one option active in that group at a time. **Multi** will allow multiple options to be active in the group at a time.

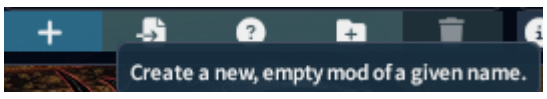
Make your mod how you want and then click **Export** at the top right of this window. It will save as a **.pmp**, Penumbra's mod package. If you wanted to add descriptions to the mod or redirect a custom path, you will have to import this .pmp into Penumbra, edit things there, and export it again through Penumbra.

For the custom path, since its folder and file have already been created for you, you can just use the *Advanced Editing* tab in your imported Penumbra mod to change the established path(s) to what you want it to be. Like if it was created as **music/choco.scd** and it's meant to replace the standard chocobo mount BGM, you'd change the redirected path to be **music/ffxiv/bgm_ride_chocobo.scd** instead, press Enter on your keyboard, and click the **Apply Changes** button above it.

If you've exported your SCDs as raw SCDs and you're editing an existing mod, find the mod you want to edit in Penumbra. Click it, and in the *Edit Mod* tab of that mod, click **Open Mod Directory**. Navigate to the file you want to replace, make a backup of it if you want, and then overwrite the original. That'd be all you have to do.

Or if you're not replacing and just adding an option, put it inside a folder in that mod. Then, in the *Advanced Editing* tab of that mod, click **Refresh Data** to force Penumbra to recognise the new file. You can add a new group and/or option specifically for that file in the *Edit Mod* tab, towards the bottom. Back in *Advanced Editing*, assuming you want your new file to modify the same sound as the mod's old one, copy the path from the other sound. Use the dropdown menu on the right of that window to select your group and option combo you made. Then, for the **Add New Path...** field below that file (should be marked in red), paste the path you copied. Click **Apply Changes** to save it.

If you haven't exported your SCDs as a full mod and instead as raw SCDs, and you're making a new mod, we'll need to create a mod template in Penumbra. Open up the plugin and go to the *Mods* tab. At the bottom left, click the **+** to create a mod. Give it a good name.



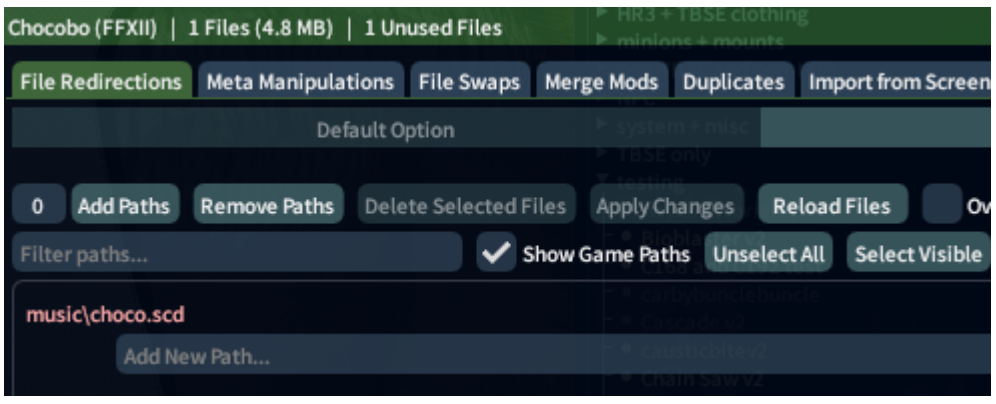
In this new mod, go over to the *Edit Mod* tab and click **Open Mod Directory**. Create a new folder in this directory. Generally, it doesn't matter what you call it or if you have a ton of subfolders within it (*C:/User/Penumbra mods/**music/never/gonna/give/you/up/***). The one thing you **must** have is one folder that FFXIV can recognise as "**valid**", with your file somewhere in that folder. That means naming the folder something like **sound**, **music**, or **chara**. If it's not there, you'll either crash or a file won't be read.

> The more you know: While you *could* make a zillion subfolders and put a file in the very last folder of that directory, it can impact read speed when your computer tries to reach that file after having to go through all those folders. It's much more obvious on hard disks than on solid state drives. This is part of the reason, if not the entire reason, that Penumbra suggests you put your mod directory as close to a drive's root folder as possible.

Place your exported file(s) in that folder (or relevant subfolder) you created.

> The more you know: If you're wondering about any .json files and if you need to edit them, the quick answer is you don't. That's all managed within the Penumbra mod. JSON files, more or less, define the editable properties, including your mod name, its description, options, and what files are going to be swapped. These JSONs also need to exist at your mod's root directory (e.g. C:/User/Penumbra mods/**new mod/**) so that Penumbra recognises that folder as a mod folder.

Now, there are three ways to go about actually implementing a file swap and making your mod work. **The first** and easiest is to establish it by clicking the *Advanced Editing* tab in your mod. You'll want to make note that it says **Default Option** in the top-right. That means, the way we're setting it up right now, when you click to enable your mod in the main Penumbra window, everything you've changed will become active by default. We'll get to the other more specific methods in a minute if you want a little more control over it.



You should see your file (or files, if you want to do multiple swaps) listed in red. If you don't, click *Refresh Data* at the top.

This is where you'll want to use the file paths you (hopefully) copied earlier somewhere. Paste the file path that you want to swap into the **Add New Path...** field below the custom path. Press Enter and you should see it turn green. Repeat for any other files you want. You can add additional paths to the same file, if you so wish. If you're satisfied with what you have, be sure to click the **Apply Changes** button and then shift back to the main mod window.

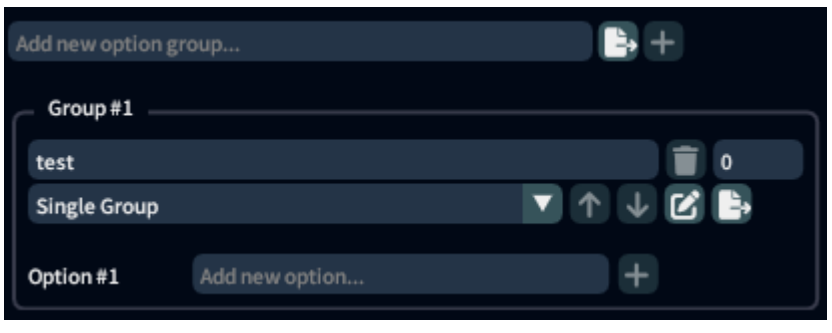
Edit anything else you feel you need to and then enable the mod. For BGM, make sure it's set in your default collection and that you're not currently playing the affected track. You may also need to click **Reload Mod** to get Penumbra to reestablish the modified paths, more so if it's BGM-based.

Your mod should be functional! :tada: You can even export it into a single file to share or save by going back to the *Edit Mod* tab and clicking **Export Mod**, which will save a Penumbra-based .pmp file in your base Penumbra mod folder.

The second option for file swapping is to allow the user to enable or disable a specific SCD without having to be stuck with the whole batch. Contrary to popular belief, not everyone wants "Anaconda" playing for every possible orchestrion. (But you know someone's going to now.) So, let's work on giving them a choice, shall we?

Go back to the *Edit Mod* tab. You'll see at the bottom you can add a new option group.

Type in what you want to name the group and click the **+**. You'll see something like this:

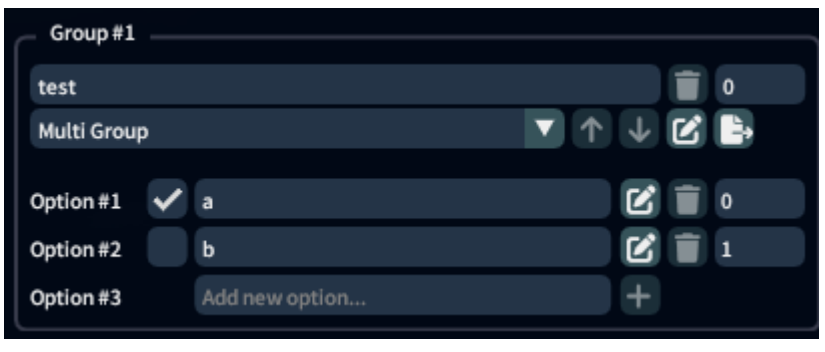


The dropdown menu below your group name will either be **Single Group** or **Multi Group**.

> **Single Group** lets you establish multiple options for the group, where only one option can be active at a time. You'd want to use this if you have a single in-game file you want to change with multiple custom files. Be aware that you need at least two options available.
>
> **Multi Group** does the opposite, letting you have multiple options active simultaneously, so you'd use it for changing multiple in-game files with a single custom file, or multiple in-game files with multiple custom files that fit into a nice category.

Note you have to click **+** for your options to actually be added to the group.

You may have noticed the numbers to the side of your group name, and next to the options if you went for **Multi Group**.



This is the priority your group and option will take over other groups and options. For most things, there's usually not much reason to have these at anything but 0 (default priority). Keep in mind that these numbers are different from the actual Priority setting in the *Settings* tab of the mod, which governs the entire mod's priority over other mods within Penumbra.

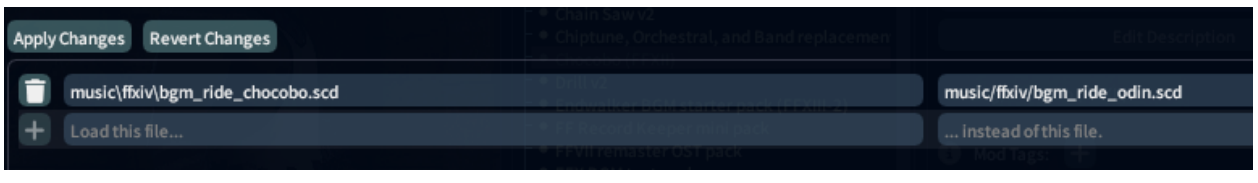
With that said, you can add as many groups and options as you want. Go ham.

Back in the *Advanced Editing* window, click on the **Default Option** dropdown in the top-right. Now you can see all the options you've added to your mod, listed as *Group: Option*. Clicking on each one will let you set new changes specific to that group/option combo. If you see a name that's gold, that's just reminding you that you've used that file already in a different group/option. You can use that file for as many options as you desire. It'll tell you on the right how many times you've used it, or you can hover over that line to find out within which options it's enabled. Additionally, ticking the **Overview Mode** checkbox tells you this same information, just in a different format.

If you went with **Single Group**, and have something like *On* for one option and *Off* for another, *On* would then be your implemented change, while *Off* would be blank without any listed changes. Anything that's left blank automatically defaults to it not being active.

The third and final file swap option is more if you just want to swap an in-game sound/BGM for another in-game sound/BGM and have no need to change anything about them otherwise. Ideally, you do this for SCDs that are similar in nature: BGM for BGM, SFX for SFX, and orchestrion for orchestrion. Swapping unlike files may result in things not playing as intended.

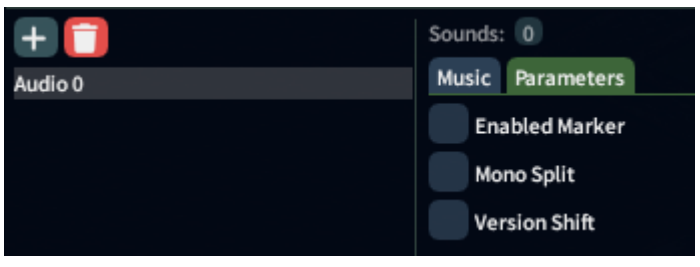
This is done in the *File Swaps* tab of *Advanced Editing* in your mod. You simply enter your desired file in the left field and then the file to swap in the right field, then you click the **+** to add it. Be sure to click **Apply Changes** to save any swaps you make. As before, this can be tied to options with the dropdown in the upper right.



A quick-ish introduction to the other SCD Editor tabs

For the purposes of keeping it relevant to this main part of the guide, I'll only go over what I believe to be the most important bits here and skip the rest. The bulkier, more comprehensive information will be in its own appendices.

Audio > Parameters



Right now, there are three checkboxes:

> ***Enabled Marker*** lets you use the *Markers* tab. This is relevant mostly to BGM that changes in combat, but can also be used in specific SFX. For BGM, the *Markers* tab allows you to set times at which the BGM transition occurs, but the file must be four or six channels in order to take advantage of this, *and* needs the **Dynamix Plus** and **DynamixStream** settings active. These will be explained in the *Sounds* tab section.
>
> ***Mono Split*** should affect mono (one-channel) audio by splitting it into distinct left and right channels. May still sound mono.
>
> ***Version Shift*** is placed mainly on SFX. As to what it does, I haven't tested it, since it's new, but it's likely related to the other *Version* fields elsewhere in the SCD.

Sounds

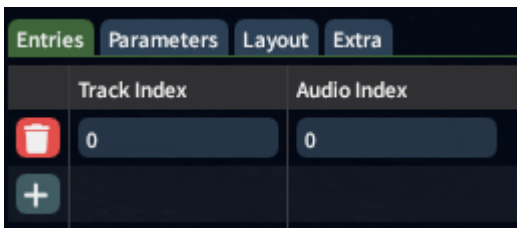
There's one field right at the top of a *Sound #* when you open it up, and that's **Volume**. It does what it sounds like it does - adjusts the volume of your file. Usually, SE sets this at 0.650 or lower (even sometimes 0.200). You'd probably want to do the same so you can still hear that you're being stabbed in-game.

For BGM, **Volume** has a minimum of 0.000 and a maximum of 1.000. You cannot go beyond those. For SFX, it's possible to go between 2.000 - 4.000. Some sound types and channels have built-in restrictions. You *can* abuse specific fields in the *Tracks* tab - check the appendix for **ADSR** - but it'll also affect other aspects of the sound. It's recommended to use an external editor like Audacity to adjust the decibel range instead.

Sounds > Entries

You've probably noticed **Sounds: 0** when opening *Audio #*.

This is so you can immediately tell which *Sound #* entry in the *Sounds* tab uses this particular audio entry. For most files, this will just be **0** since there's normally only ***Sound 0***. Specifically, it's determined by the *Audio Index* value here, in the *Entries* tab under *Sounds*:



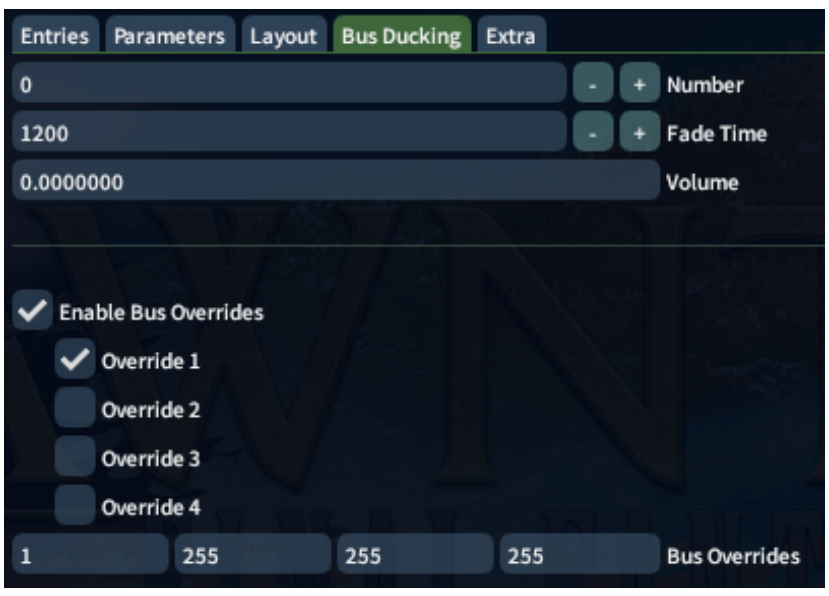
Track Index then indicates which *Track #* entry in the *Tracks* tab that *Sound #* uses. For most common sounds, this will just be **0** as well. If you open up a file like *sound/vfx/SE_VFX_common.scd*, though, you'll quickly notice how many entries there are in each tab, to which *Entries* will help figure out how they all tie together. Because they can't just be in order, apparently.

Sounds > Parameters

Loop, while checked, will flag your file as having a loop; if it's not a looper, leave it unchecked. It's possible that a file can still loop without this flag, as long as it has the *Loop Time* metadata set, but I'd check it anyway to be on the safe side. *Be careful about applying this to music that only plays once*. I'm talking mostly about early ARR dungeons that transition into silence after the BGM is done. (Specifically, it'll call the *music/ffxiv/BGM_Null.scd* track because that's how it's defined in one of the game's EXD datasheets.)

There's a **Music** checkbox, but I'm telling you right now it has no effect and is never used in FFXIV. Even BGM doesn't toggle this. You're welcome to check it if you really, *really* want to, but it won't do anything (except maybe in other SE games like FFXIII, but I haven't tested that). Same with **Music Surround**.

Bus Ducking allows a specific system channel like BGM or voices to change in volume, most often so it doesn't play over any other system channel. Checking this will create a *Bus Ducking* tab that you can click on:



Number determines which bus number's volume to affect. **0** will make it read the *Bus Overrides* section instead. You can check or change which one your SCD uses with the *Bus Number* field back in the *Parameters* tab here. As a quick reference, here's a handy list of bus numbers used across the board:

General audio bus numbers:

(undefined): **0**
music: **1**
skills, emotes, cutscene audio: **2**
monster sounds: **2 / 3**
voices: **3**
SE_UI, minigame sounds: **4**
objects (housing, combat, instances): **5**
ambient/background: **5 / 6 / 7 / 11**
footsteps: **6**
limit breaks: **6 / 7**
*placednpc_*: **7**
weird niche BGM cases (*TimeStretchBGM*): **8**
system sounds: **9**
SE_VFX_common: **10**
walla/crowd sounds (*Gaya_*): **12**
title movie tracks: **13**
[unknown]: **14**
[unknown]: **15**
orchestrion: **16**
bard performance: **17**
vibration sounds: **18**
[unknown] (marked as *Housing* in ClientStructs): **19**
*randomwind_*: **20**

Fade Time is how long it takes to reduce the volume of *Number* or the *Bus Overrides*, measured in milliseconds.

Volume specifies the ending volume the ducked audio channel(s) should have. This can be anywhere from 0.000 to 1.000. In case it wasn't obvious, make sure *Number*/*Bus Overrides* and *Bus Number* in *Parameters* aren't equal if Volume is **0.000** or you'll just mute your file altogether (unless that's what you want).

Bus Overrides essentially work in place of *Number* to allow muting of multiple bus numbers instead of just one. You can read more about this in the detailed Sounds > Parameters section in Appendix B.

For an example of bus ducking: pre-ShB orchestrion BGM, by nature, will have the values of **1** for *Number*, **1200** for *Fade Time*, and **0.000** for *Volume*. This means any BGM playing (or anything else that shares that bus value of **1**) will be muted (volume at **0**) after **1.2 seconds**, which is why something like the Orchestrion plugin won't play over any current orchestrion music, because all the songs have a bus of 1. The irony is real.

Acceleration determines how sounds will change volume according to movement. This is normally applied to mounts. When checked, it creates a corresponding subtab of *Acceleration*.

Version, whether at the top or in each dropdown, is currently unknown. I'd just leave it at **0**.

Acceleration Count is normally set to **2**, meaning the SCD will only pay attention to the first two *Acceleration #* entries. (Note that these entries cannot be manually added yet.) The **first entry** pays attention to **when you stop moving or are idle**, and the **second** notes **when you move**.

Volume in *Acceleration #0* is the overall volume when you don't move, while **Volume** in *Acceleration #1* is the target volume when you move. As an example, setting the **first** *Volume* to 0.000 and the **second** *Volume* to 1.000 will mute the BGM when you're not moving and make it full volume when you do. These fields act *proportionally to the volume of the sound in general*, so if your main volume in the *Sounds* tab is 0.200, having any *Volume* in *Acceleration* at 1.000 will *max out at 0.200*.

Up Time is used by *Acceleration #0* and **fades in** for that set millisecond time, according to its *Volume*. **Down Time**, conversely, is used by *Acceleration #1* and **fades out** for the set time, according to its *Volume*. The opposite field won't be read if swapped, only the one that was meant for that particular *Acceleration #*.

Extra Desc, when checked, will create an *Extra* tab.

One purpose of **Extra Desc** is to affect overall frequencies slightly by damping sharper sounds for a subjectively better auditory experience. (You can get a feel for this if you remove *Extra Desc* on a SAM skill.) In addition, DualSense rumble and audio can be specified in this tab. Almost all of the more recent sounds in-game use *Extra Desc*, including BGM. This is, by no means, a required flag, but may be something to consider using.

Version defines which version of the *Extra Desc* code to use. Earlier sounds have **0**, some later sounds have **1**, while Dawntrail sounds will have **2**. There's a slight difference in how the sound plays according to version, so test them if you wish, but my recommendation would be to use their latest versions, **1** or **2**, since they're theoretically an improvement.

Unknown 1 is a reserved field in case they add something in the future. You can leave this at **0**.

Play Time Length is where you'd want to insert your file's *length in milliseconds*. For looped files, this extends only to the end of your loop. For multiple sounds using the same *Sound #*, it's the *length of the longest sound in the group*.

Unknown 2 has several possible uses and values. While some of these are currently unknown, a known value of **2** will make any connected DualSense controller play sound from its speaker and contribute to its rumble feature. The sounds for class change, job gauge becoming full, and footsteps are examples of this use. (Thanks, Ashley, for testing!) **32** will damp frequencies. Use **34** if you want both DualSense and frequency damping. For other values, see Sounds > Parameters in Appendix B.

Unknown 3 I would leave as is, since this is still unknown.

And now for the main section in *Parameters*:

Bus Number, as discussed in the **Bus Ducking** section, is what system channel the SCD will use to play its sound. The most common ones you'll see are **1** for BGM, **2** for SFX, and **16** for orchestrions.

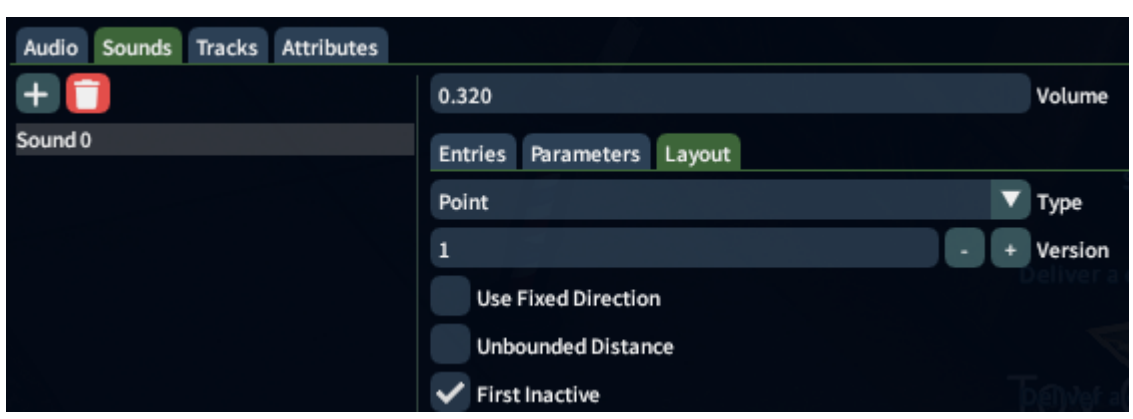
Priority functions like you'd think. If two SCDs of the same *Bus Number* play simultaneously, it then (should) fall on *Priority* to determine which should play over the other or take priority on its metadata.

> Side note: For the most part, *regardless of this setting*, there can only be a certain amount of unique sounds at one time per system channel, which is why a lot of combat SFX will end up cancelling each other out. There's a way to circumvent this on a sound-by-sound basis, at least.

Type is generally how the sound will behave and has the potential to unlock specific metadata to support that behaviour.

> The most common type is **Normal**. This is the catch-all type, where you don't need anything really specific happening, just what's provided by default.
>
> **Invalid** is something you shouldn't see unless you've just created a new *Sound #*. If it's set to this, you may want to make it **Normal** so the sound actually works.
>
> **Empty** is somewhat common, and just means that there's no audio to be used for this particular *Sound #*. While it may seem like an oxymoron to have sound but no sound, it still adheres to any metadata like *Reverb* or *Bus Ducking*, so it has its niche uses.
>
> **Cycle** is mostly for ambient SFX. It uses sets of sounds to cycle through at certain times. An example of this would be the thunder cracks you might hear while it's raining, or the birdsongs during the day.
>
> **GroupRandom** determines which sound should play from a defined group of sounds in the SCD, and uses a *Limit* field to determine % chance of each. As of writing, this unfortunately cannot be applied to randomise BGM, unless the game allowed it overnight. Someday, perhaps.
>
> Finally, **DynamixStream** is what allows a BGM to transition while in combat. This is for stuff like PvP music, Bozja, pre-Stormblood alliances, or some hard-mode dungeons. The combat music out in the field - like if you run into something in Il Mheg - is a separate file and doesn't utilise this type. Note that there are some caveats to **DynamixStream** in order for it to work - see Appendix B for more info.

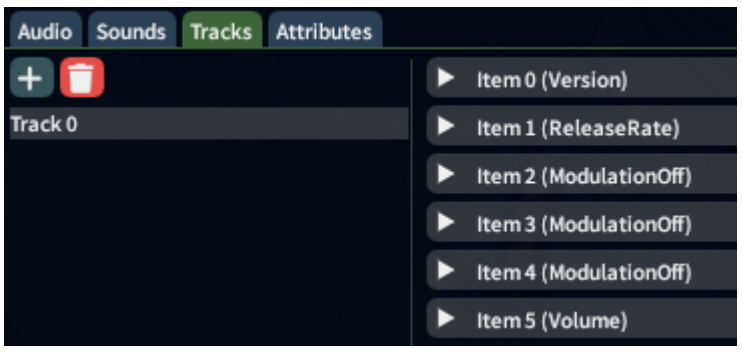
Sounds > Layout



We don't need to concern ourselves with a majority of this, unless you're heavy into audio edits, and, even then, it's mostly with SFX.

> **Volume** (the one with **four values**) scales off of every other volume setting. Only the first value makes an actual difference in standard gameplay. There's probably a reason for the others (axes, perhaps?), and maybe it's to do with the Immerse sound pack thing, but I've no idea.
>
> 
>
>
> **Volume** (the one with the **single value** in the *Layout* tab, not at the very top) scales off of every other volume setting. Not too much reason to change it. Like most other volume settings, this can only hold a value between 0.000 and 1.000.
>
> 
>
>
> **Pitch** will adjust the overall pitch of your sound. This can be above 1.000. It will also timestretch it. Does not affect BGM. There's a pitch value in the *Tracks* tab as well, so be aware of that if you find your sound still seems higher or lower than normal. **Do not set any Pitch value below 0.000**. You will crash.
>
> **Fade In Time** and **Fade Out Time** relate to how long it takes for the audio to fade in or fade out when swapping to or from another audio track. Still, though, having them at the default 2000 (milliseconds) is perfectly fine.

Tracks



First off, you can change what each of these items will do by opening them up and clicking the **Type** dropdown inside. Additionally, you can add or delete items if the current ones aren't sufficient. The problem lies in how FFXIV wants to read these fields and what general order they're in. Always remember what you changed and what the original was, in case everything explodes.

> **Volume** and **Pitch** work the same as they do elsewhere. **Pitch** can be above 1.000. **Volume** cannot. Below 0.000, bad things happen, like crashes. We don't like crashes.
>
> **Panning** will pan your sound to the left or right. **Between 0 and -2** will pan to the left, while **between 0 and 2** will pan to the right. Keep this at **0** if you don't want to pan anything.
>
> **RandomVolume/RandomPitch/RandomPan** work as described. The **Upper** value is the maximum it's allowed to go, while **Lower** is the minimum. If you swap the values, you'll crash, so be sure **Upper** > **Lower**.
>
> **ModulationOff** I'm still not sure, but intuition tells me it helps to reset the **Carrier** specified (tied to **Type**) on each successive call of that *Track #*. Probably best to have these here in case.
>
> **Interval**, as previously explained, helps to establish loops or prevent sounds from cutting off early. It can also define how long to do something or wait before doing something else, among other things. Always measured in milliseconds.
>
> **RandomWait** works somewhat similar to *Interval*, but lets you randomise the length. **Value 1** is the shortest amount of time to wait, while **Value 2** is the longest amount. This is most often found in ambient files, like spacing out raindrops at random times so it seems more natural.
>
> **KeyOn** is mainly to establish *Interval* items that relate to total length in some way. There can be other cases, like putting in *Pitch* or *Volume* in between *Interval*s to modify them at specific times. But after setting all values that are needed here, it needs to be followed by one of three items: **LoopEnd** for loops, **KeyOff** for most other sounds, or **CutOff** for specific cases. **CutOff** is (probably) used to end a sound prematurely if something interrupts it, like a cutscene or a keypress during voiced dialogue with a text window. I'm unsure if **KeyOn** *needs* to exist in general, but something tells me it does.
>
> **End** is always at the very end of all items. Be sure to include it.

In some cases (probably not with BGM, just SFX), you may find an **Interval** value established before any **KeyOn**. This is usually accompanied by its presence in a group within the *Sounds* tab, and would then dictate the millisecond time at which that sound would play, using the **Play Time Length** value in the *Extra* subtab as the base length (equivalent to the longest sound in that set). The reason this exists is so specific parts of the sound can carry modulations like **Pitch** and **Volume**. You're most likely to find instances of this with weapon draw and sheathe sounds (e.g. book draw, book open, and then a special SFX for unique ones).

Attributes

This tab is relatively technical and generally has no bearing on common sound uses, so we can ignore it at this juncture. If you're curious, check Appendix E. I also have done moderate research on what shows up there in general, which can be found in a separate document listed in its own section.

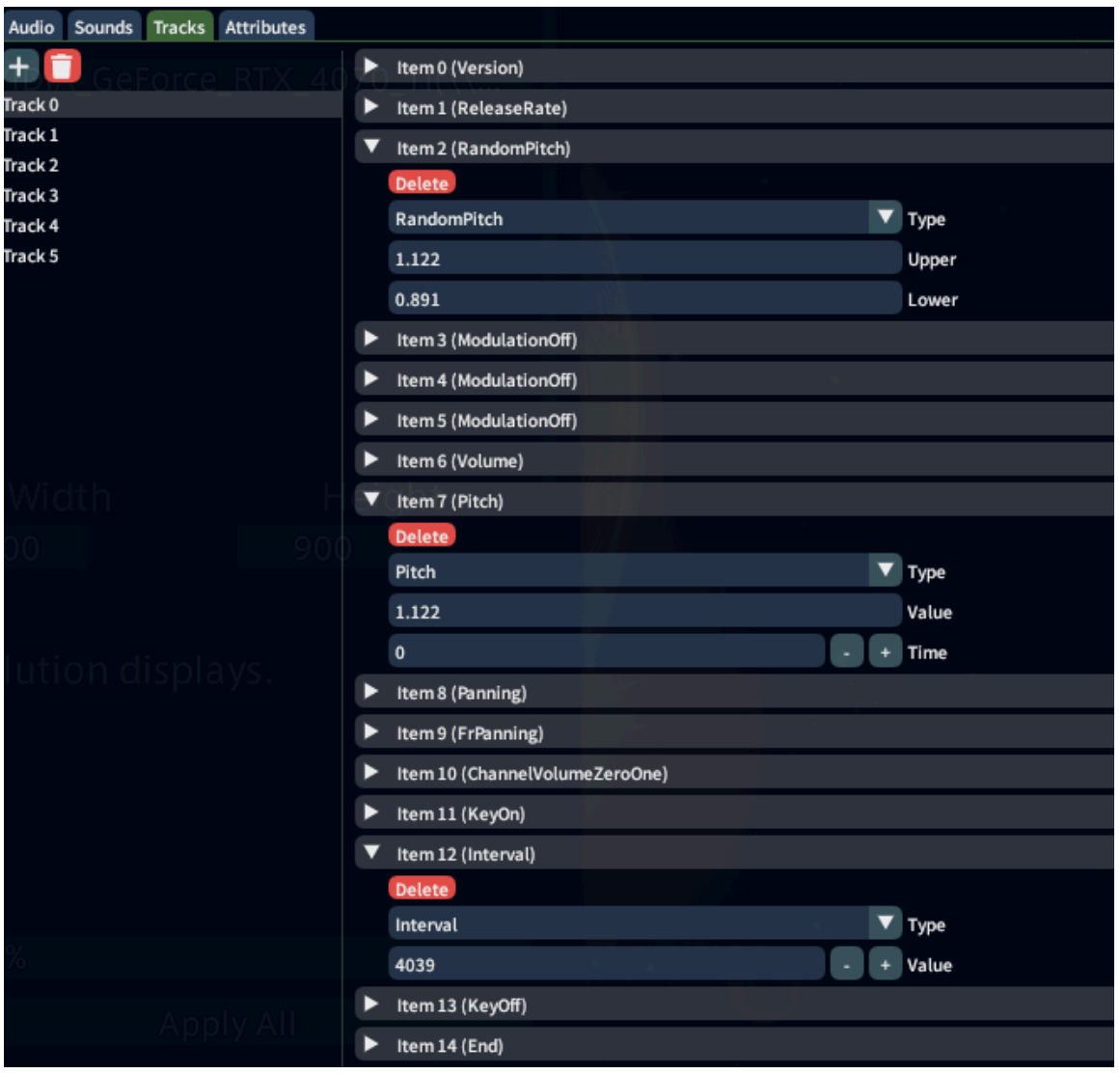
Appendix A: SFX swap and modulation for mounts (and weapons or items)

This section describes mount sound manipulation, but the same general process can be used for weapon and item SFX with multiple audio entries, as long as you know what your original sound path is.

Let's use **sound/battle/mon/3854.scd** as a running example - the magitek death claw sounds. Say you want that buzz-like sound that occurs when you move to be on a different mount. (Hi, friend that I helped with this exact same thing.) In this case, it's listed as **Index 21**. You'll now want to look for the index number under the ***Sounds*** tab, which is easier said than done when there are almost always 29 listed. A recent VFXEditor update *does* show sound indices when checking *Audio #*, but only *sometimes* (related to something called *Limit*). This is a minor bug, so for now, we'll have to do it the hard way.

Thankfully, most of these *Sound #* entries should have a **Type** of **Empty**, so you can skip searching those for relevant metadata. For the others, you want to check the *Tracks* subtab and look for the matching value in the Audio Index field. To save you time on this one, it's **Sound 12**. Keep note of the **Play Time Length** value in the *Extra* subtab - we'll get to that soon. Here, it's **4039** (measured in milliseconds). The more important value is that within ***Track Index***, which is referenced directly in the main *Tracks* tab. Here, it's **0**.

Moving to said *Tracks* tab, we can now check Track 0, as mentioned. You'll see there's **Item #2 (RandomPitch)**:



Upper tells us how high the pitch of the sound can be at any given play of it, while **Lower** tells us how low. You can, of course, change these as you will, so long as Upper has a higher value than Lower. **Item #7 (Pitch)** seems to designate the base value for the sound. While some files have this equivalent to the Upper value, others have it set to **1**. As always, the safest bet is to match the original.

If you open up one of the items marked **(ModulationOff)**, there are quite a few options to pick if you click the dropdown inside of it. Many of these are untested so far. Mostly, you just tend to see RandomPitch used. I'll go into further detail at another time, once I've discerned what more of them do.

For right now, make note of the **Upper** and **Lower** numbers (and the **Pitch** value, by extension), as we'll be doing a sound swap for testing soon. Go back to the main *Audio* tab and extract **WAV** (as this is a MsAdPcm file).

Now open up **sound/battle/mon/13135.scd** - sounds for the lunar whale mount. Assuming we want to change its movement sounds, these will be **Index 4 through Index 6**. (Note that it won't be the same every time, so you have to check around.)

This is more of a rough-and-ready solution, but I chose to replace all three with the same file that was extracted just a second ago. They all deal with movement, but there's a slight difference in when they're used. Anyway, we'll do the same thing with finding audio indices in the *Sounds* tab.

Here's where you want to be a little careful - some of these sounds are used more than once. For this mount specifically, our corresponding entries are under **Sound 12** and **Sound 14**. Same as before, note each track index. Don't worry about the Limit values; they doesn't need to be changed, so best to leave it as is. You'll additionally see *Track Index* #3 has a very large value for its *Audio Index* (20479). It's likely related to Track 3 having (PlayInnerSound) and a bank number, but you're safe to leave and ignore it.

Let's shift over to the main *Tracks* tab. The ones we want to edit will be **Track 0, 1, 2, 4, 5, and 6**. Should this be applicable in the future, we'd want to change our **Item #11 (Interval)** to reflect our replaced audio's length (**4039** in this example) for each Track entry. The Item # might be different for other mounts, but just remember the one you want to modify will be between *KeyOn* and *KeyOff*.

Since we probably want to retain the original mount's pitch modulations, let's go ahead and add those. For each track, open up the first **ModulationOff** and switch the dropdown to **RandomPitch**. This should look familiar, so go ahead and input those values you wrote down earlier. Additionally, where you see **(Pitch)**, change that to the **Upper** value.

While I'm unsure if this makes a difference (wow, I'm really unsure of myself in general, huh), I like to change the other *(ModulationOff)* entries to have **Pan** and **Volume** respectively in their Carrier dropdowns if those are unused elsewhere, just to be safe. Likewise, if you modify a pan effect and don't use the other two, change those Carrier entries to Pitch and Volume. Same deal with Volume changed. The one thing you *don't* want to do is have those at None. Absolutely do not do that unless you like crashing.

Now we can export the raw SCD and use it (or slot in the sound to be replaced and export as a mod, if that's your preference). Your lunar whale mount should have the same movement sounds and effect as the magitek death claw, save for what we didn't modify. Note that this might not work if you've modified your lunar whale (or whatever mount you changed) to use a different PAP or AVFX. I spent a bit of time trying to figure out why that is, but it's a little out of my depth. My guess is something's missing or there's an overwriting sound. But for vanilla mount purposes, it should be functional.

Appendix B: Audio / Sounds > Parameters

Audio > Parameters

Enabled Marker

Does what it says - enables the use of markers and creates a *Markers* tab. This mostly has known use for dynamic combat BGM, to tell the game at what times to shift it when combat status changes. However, some skill sounds will also choose to use this. It's unknown at this time what the latter one does, or why it can be 0 or some incredibly small value. Could be related to how *Version Shift* functions.

Mono Split

Possibly acts to change a mono sound into separate stereo (left/right) channels. Generally doesn't seem to be used. I believe I saw this once or twice, but it's a bit rare.

Version Shift

This is also a little bit of an unknown, although a good number of sounds use this setting. Maybe has something to do with all the *Version* fields you find in an SCD. The question is which one and what the purpose is. Let me know what you find out!

Audio > Markers

This tab is created as a result of checking the **Enabled Marker** flag.

Id

ID only ever seems to hold the string "**MARK**" as a way to define that it's being used for marker positions.

If the **Id** field is blank, you will crash when you play the sound.

Loop Start / Loop End

These fields are ways to establish the loop points for the marker set. It's possible this acts independently of other loop values and will continue the sound at the given *Loop Start* value after its interruption, while *Loop End* is for the marker stop point only.

For example, in Labyrinth of the Ancients, you'll notice the BGM doesn't begin from the start again after you finish a boss battle, rather at a set point, even though it's using the same SCD and audio index.

If you don't want to use loop values, just set both to **0.000**.

The more you know: In *Dynamix Plus*-type BGM, markers are, on average, two to three seconds apart, landing at the start of each measure. You may also see marker values go beyond the determined *Loop End* value or have several repeating values after it. I don't really know why this is, but it seems likely it's some scrapped plan for the sound.

Below *Loop Start* and *Loop End*, you can add or remove marker values. These are measured in seconds.

To do: test more on markers for combat sounds. These usually don't have loop values and just one or two values for markers that run sound length.

Sounds > Parameters

Side note: If you're working with BGM, you'll likely only see a single *Sound 0* collection. Anything else and you'll get quite a few *Sound #* entries. These help to group multiple sounds for a single action, or it can occasionally carry an argument on its own with no sound attached.

Volume

Sets the main volume for the SCD. This has a soft maximum of **1.000** and a minimum of **0.000**.

This can be set higher than **1.000** in certain situations - mainly SFX, which can mostly be 3.000-4.000 at max - but I haven't done a lot of research into which ones and why.

BGM and orchestrion are capped at **1.000**.

The sounds on bus 18 (chiefly, for controller vibrations) are hardcoded to be a little more amplified, so you can leverage this, if you want.

Loop

Flags the file for looping. This can sometimes not hold weight for if a file actually loops, so it may just be a failsafe or referenced by some other field.

Reverb

This should apply reverb to your file. I can never personally tell if something has that, so I'm just guessing that's correct. It'd be odd if it weren't, but I've certainly been proved wrong many-a-time.

Fixed Volume and Fixed Position

These two checkboxes work in tandem with in-game objects that don't move on their own, and do as their name implies - keeping volume the same and playing the file only at the position it was placed. These are both checked for most dedicated cutscene SCDs.

Fixed Position, in particular, can help to even out volume when the camera's panned around the source.

Music

This doesn't do what it says on the box. This is never ticked for any music files. In fact, I've never seen it used (except in custom BGM SCDs in dance mods, but it's pretty much a pseudo effect). Ticking it with an SFX doesn't break my game, so, I don't know, go ham.

Bypass PLIIz

This is assumedly concerned with bypassing Dolby Pro Logic IIz and part of its surround-sound capabilities. The only places it's been seen so far are with the generic air whoosh sound you hear while flying a mount, some cutscene SFX, and the *_randomwind* sounds for areas with large grassy fields.

Examples:

sound/battle/etc/SE_Fly_AC_Sound.scd

bgcommon/sound/bg_ex01/bg_ex01_randomwind_grass_a01.scd

Testing with this, it's possible to manipulate how many concurrent sounds from the given *Sound #* will play, as well as which sounds from outside of the affected *Sound #* (some weird combination of *Unknown 3, 4*, and *6*, I guess?), assuming the *Sound #* is *GroupRandom* or *Cycle*. It can also change the perceived volume of the attached sounds and even which *Track #* it draws from. A lot of these values rely on others in this tab to function, so this will require more thorough testing.

Use External Attr

This could be tied into the *ExternalId* attribute tag. *If I could find it being used*.

Exist Routing Setting

Clicking this creates a new subtab called *Routing*. Currently it's marked as **[WIP]** and has some filtering/EQ stuff to select from, along with an adjustable filter count. Filter Count > 1 will kind of muffle the sound, no matter which *Type* is selected.

Music Surround

Probably does what it says and applies a surround effect to music. Not that I've seen it exist or particularly hear a difference, but I also don't have a surround setup.

Bus Ducking

When checked, this will give you a *Bus Ducking* subtab. There's quite a bit going on, but you likely don't need to worry about changing most of these values, unless you want to have greater control over how volumes are adjusted across bus numbers.

Number

Setting this at any value other than **0** will determine what singular bus number will be affected for the length of your file. If your file is looped, the target bus will be affected until that loop is manually terminated.

Number is the default field used by pre-Shadowbringers SCDs; this has since been superseded by the *Bus Override* fields in this tab to cover multiple buses. You can still use only *Number* to keep it simple, but it's recommended to uncheck *Enable Bus Overrides* if you do this.

A *Number* of **0** will automatically defer to *Enable Bus Overrides*.

Fade Time

Sets how long it takes the affected sound to reach the *Volume* set in the field below it. It's used for both fade-out and fade-in time, but separately. This is measured in **milliseconds**.

For example, if *Fade Time* is **1000** (one second), it will take one second to fade out, wait for the sound to reach its end, and then take one second to fade in.

Volume

This determines the ending volume to use on the affected bus after the duration set in *Fade Time*.

This has a maximum of **2.0**. Negative values will default to **0.0**.

Enable Bus Overrides

Allows the *Override 1 - 4* checkboxes to function, which, in turn, allow the *Bus Overrides* values to be read.

Unchecking this will let *Number* take precedence.

Override 1 - 4

The checkboxes correspond to the four values below them - with *Override 1* affecting the leftmost - going in order.

Bus Overrides

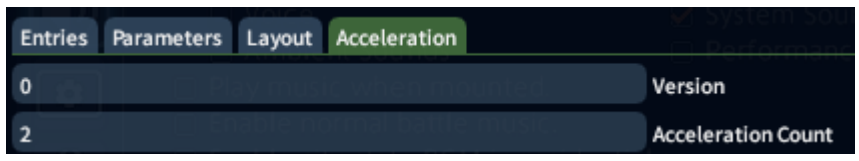
Specifies the bus numbers that are affected by this tab if *Enable Bus Overrides* is checked. Up to four bus numbers can be inputted at this time.

If any numbers are not being manipulated, they are set to disabled: **255** (-1).

It isn't required to enable these numbers subsequently - i.e. you can have *Override 1* and *Override 3* active, and skip *Override 2*.

Acceleration

Governs mount behaviour for stop/start music volumes. Creates an *Acceleration* subtab. Note that, in the current version of VFXEditor, adding the *Acceleration* flag does not automatically include the relevant acceleration fields, so you'll need to find one that already contains this data, such as mount BGM.



Version

Unknown what it does, but if it exists with a number already in your base file, it's best to use that. Also, let me know if you see anything other than **0** on a vanilla file, thanks!

Acceleration Count

This field's value is usually set to **2** - one for *Acceleration #0*'s velocity increase and another for *Acceleration #1*'s velocity decrease. While four total *Accelerations* exist, I don't know how the other two function or where that would be used. If you find anything beyond two, absolutely let me know.

Version

Unknown. I haven't seen this at anything other than **0**, and there's been no real reason to change it so far.

Volume

Sets the amount your sound will increase (when used in *Acceleration #0*) or decrease (when used in *Acceleration #1*).

Up Time / Down Time

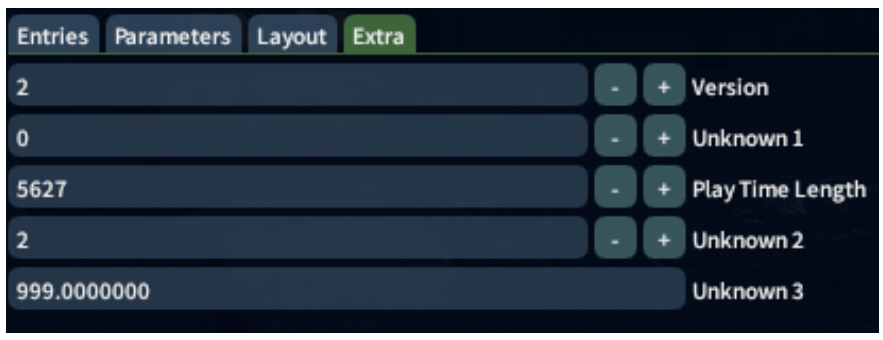
Up Time sets how long your sound will take to increase, while *Down Time* sets how long it takes to decrease. These are measured in milliseconds. By default, they're set to **0.300** and **1.000**, respectively, with any volume adjustments made in the main *Sounds* header instead. If you plan to change the acceleration volumes themselves, it's recommended to only change *Acceleration #0*'s *Up Time*. *Acceleration #1*'s *Down Time* you should probably leave at **1.000**, since it works at a percentage of your *Acceleration #0* value.

Dynamix End

Unknown. I haven't seen a file use it, not even in conjunction with one that has **Dynamix Plus** active.

Extra Desc

Allows for specifying some additional data for the sound.



Version

This can change how settings in this tab can affect the sound. This makes the most difference with *Unknown 2* and its sound damping capability.

Vanilla files currently use a *Version* between **0** and **2**. It's recommended to stick with the most recent version used in Dawntrail (**2**), unless you have a specific reason to not do so.

Unknown 1

Unknown. This is sort of a reserved field, just here in case they decide to use it in the future, so should just be set to **0**. Let me know if something different shows up or if you get a value to do something.

Play Time Length

Sets how long, in milliseconds, that the data in this subtab will run from the start of the sound. In almost all cases, this will directly match the length of the entire associated audio (e.g.: if the audio is 7.34 seconds, *Play Time Length* will be **7340**). In practical testing, this value doesn't seem to matter, although I'm sure in some regard it does.

Unknown 2

This has several possible uses and values. I'll change this into several checkboxes soon-ish, but for now, if you want to use multiple, add the values together.

1 - unknown. (Used in combination with other settings. This'll make more sense when it turns into a checkbox.)

2 will make any connected DualSense (and DualShock?) controller play sound from its speaker and contribute to its rumble (vibration) feature. The sounds for class change, job gauge becoming full, and footsteps are examples of this use. (Thanks, Ashley, for testing!) Performs a similar function to **8**, but with a slightly stronger vibration. Both **2** and **1** are used by some DoL skills, as well as for materia-attaching sounds.

4 will mute the sound in the application while playing it (or, at least, a version of it) on a controller's speaker. Tested with DualSense. Note that this only occurs if a compatible controller is connected; otherwise, you'll still hear the sound in-game. This will use a slightly less intense vibration. On some sounds (like system sounds), an additional sound may play directly after.

8 is similar to **2** in that it provides vibration and speaker capabilities for the DualSense. The vibration can be slightly less intense than **2** with some sounds. May be equivalent, depending on how strong the lower frequencies are. Some "zingles" (like *GS_Ride_Countdown* [gold saucer's Air Force One countdown]) and fishing skills will use **10** (8 + 2).

16 is used by weapon sounds, particularly when a weapon is swung or makes contact with an object. There seems to be a very subtle change in the sound, though I'm unable to say exactly what it is.

32 applies a high-shelf filter, meaning that higher, sharper frequencies will be slightly damped. (Thanks to Hoshi for the initial discovery.) This is more obvious when tested on something like samurai skills (Higanbana's loop SFX or Hakaze). This value is most often used by ranged skills. Melee skills will use this and **16** (so a total value of **48**).

128 is used by *_randomwind* SCDs, which exist in large, grassy plains like in Othard, in Mist housing areas, or even the Barbariccia fight, if you lower the BGM enough to notice it. Unknown what this setting does.

Unknown 3

Unknown. So far, it's either been **0.0** or **999.0**.

Dynamix Plus

Used along with a *Type* of *DynamixStream*. This flags the file as being able to reference a different audio channel set within its host sound. For example, if you have a six-channel sound, it'll reference them in three distinct groups: channel 1 + 2 for base audio, channel 3 + 4 for special-case audio, and channel 5 + 6 for transitional audio.

This is basically used for things like PvP, early alliance raids, and Alexander: The Soul of the Creator phase two (A12), where the BGM will dynamically change according to combat status. Note that the actual capability is set externally in one of the game's spreadsheets; enabling *Dynamix Plus* and *DynamixStream* will just determine how the SCD will behave if and only if the spreadsheet says so.

> Testing note: I've done several tests on replacing separate channels and exporting different channel numbers. Doing the standard six-channel audio export in Audacity (which you can enable in *Preferences > Import/Export > Use custom mix*) and importing will register all channels (meaning you can see it labelled as **Vorbis / 6 Ch** in VFXEditor), but it mutes the entire game's audio. It remains like that for a good chunk of time after changing BGM. Importing as three-channel won't play it, but game audio is fine. Importing as two-channel treats it like you just did a straight replacement: it ignores combat status and, naturally, shoves the music into left/right channel. Importing as four-channel works fine so far, but lacks the natural transition six-channel has. Multichannel support is still being ironed out.

Atomosgear

This creates its own sub-tab and correlates to specific SFX that rely on a set amount of player characters to exist in an area in order to play, such as ambient crowd sounds. This checkbox only applies when attached to *sound/strm/gaya_** files; this is likely defined externally in a spreadsheet (like a special condition field).

Version

Unknown. Always at **0**. May increase, depending on if they revise how *Atomosgear* functions.

Minimum Number of People

This is the minimum threshold for how many players need to exist in an area before it starts playing.

Maximum Number of People

Sets a maximum for player count in a given area, and then does a calculation to determine a percentage-based volume increase: *Minimum Number of People* ÷ *Maximum Number of People* = **volume %**. I might need to test this further to make sure it's actually volume increase and not referencing the specific channels of the audio.

Bus Number

Bus numbers range from 0 to 20 so far. In general, each subset of sound is routed to a specific channel, which allows careful overlap of multitudes of sounds and the ability to reduce (or increase) a certain channel's volume as necessary without affecting the others.

It should be noted that **0** is a special case in terms of *Bus Ducking*. This means it'll defer to the *Bus Overrides* information below it in order to specify up to four specific bus numbers to affect. An example would be banners that have sounds attached, like those for quest accepts - it has a base bus of 9, but uses the *Bus Ducking* flag with a number of 0 and then *Override 1* with a value of **1**. You kind of could just do the name by leaving *Number* at **1**, but you could consider the override method *de facto* for SE's purposes since Shadowbringers.

Additionally, bus number **8**, in particular, is for time-stretch BGM, most notably (or only?) Thornmarch phase one, where the track speeds up on each downed enemy. If modified to **1**, this behaviour no longer happens, but the reverse cannot be applied in normal situations. There's a definition somewhere outside of the SCD (perhaps an EXD file) that determines this. Handy chart of general numbers (a list can also be found in a decompiler):

General audio bus numbers:

(undefined): **0**
music: **1**
skills, emotes, cutscene audio: **2**
monster sounds: **2 / 3**
voices: **3**
SE_UI, minigame sounds: **4**
objects (housing, combat, instances): **5**
ambient/background: **5 / 6 / 7 / 11**
footsteps: **6**
limit breaks: **6 / 7**
*placednpc_*: **7**
weird niche BGM cases (*TimeStretchBGM*): **8**
system sounds: **9**
SE_VFX_common: **10**
walla/crowd sounds (*Gaya_*): **12**
title movie tracks: **13**
[unknown]: **14**
[unknown]: **15**
orchestrion: **16**
bard performance: **17**
vibration sounds: **18**
[unknown] (marked as *Housing* in ClientStructs): **19**
*randomwind_*: **20**

Priority

This works together with *Bus Number* and *Attributes* to allow a sound to overlap, cancel another out, or naturally reduce its volume, depending on its value and bus placement.

A somewhat simple example: You might have noticed that in alliance raids, a lot of skill sounds will override others when played quickly. Setting up an *Attribute* can alleviate this a bit by increasing allowed repetitions, but if the *Bus Number* and *Priority* values are the same as other sounds, things may still end up being cut short. A higher priority should give an SCD priority in this case, so that the sound won't be affected by others in high-volume situations.

While it may sound like I'm confident in these assertions, I am extremely not and completely guessing. <laughs> *I'm so predictable*.

Type

Defines how the affected sound group as a whole should be handled.

Side note: You may occasionally come across an *Audio Index* value in the *Entries* tab that doesn't correlate; it'll show as **20479** (two-value split: **-1** and **79**). This means instead of defining an audio index directly, it'll defer to the track index's *PlayInnerSound* item for what to do instead.

Normal

This is the default *Type* for most sounds and means the sound group will behave normally. How normal.

Invalid

Setting something to this will make the sound not play. It's recommended to use *None* instead.

Random

This can randomise which sound is played. This is seemingly never used in FFXIV; instead, they opt for *GroupRandom*. However, sound files such as those found in FF13-2 can contain this *Type*. Note that in those files, the secondary entry in *Entries > Limit* remains at **0** for each one, using only the percentages:

Stereo

Not used in any current vanilla files. This doesn't do what it says, or at least on my test file that's an MsAdPcm mono; it just mutes it.

Cycle

This is more or less reserved for sound effects. The SCD will have sounds that play at set intervals which are placed in the main *Tracks* tab. There may sometimes be a base file included that plays along with it. Sometimes this is in its own SCD. The *Tracks* subtab in *Sounds* contains preset entries for each available sound in the SCD. *Cycle Interval*, *Cycle Play Track*, and *Cycle Range* are unknowns at this time. *Track Index* corresponds to the main *Tracks* tab entries, while *Audio Index* corresponds to the *Audio* tab entries. *Limit* defines % chance of audio play.

Order

This might be to play each sound directly after each other in regard to audio index. Not currently used in any known files.

FourChannelSurround

This most likely follows its namesake, to mark a file as 4.1 surround.

Engine

Unknown. Feel free to test it and let me know if something happens.

Dialog

This is unrelated to actual voice files. That's all I have for you.

FixedPosition

Seems like a weird one. That flag already exists under attributes below, yet, even when it's checked, this Type isn't selected. TBD.

DynamixStream

Rehashing what was mentioned in the *Dynamix Plus* heading, this relates to six-channel BGM that switches between combat and out-of-combat status. You can replace the file with something that isn't six-channel and it won't break, but you also won't receive the benefit of this *Type* selection unless it's four- or six-channel and the appropriate data is defined in an external datasheet.

GroupRandom

Sets a random sound play from a group of sounds. In preset files, the *Tracks* subtab will have entries for each possible sound. *Track Index* and *Audio Index* are as usual.

For *Limit*, the first field is % chance that the track will play on each call of the SCD, but it's more of a "if the lowest chance fails, try the next one" situation. In other words, limits of **30**, **70**, and **100** mean that the first has **30%** chance, the second has a **70%** chance if the 30% fails, and the **100%** will play if both 30% and 70% fail.

The second of these *Limit* fields - the **0, 1, and 2** in this current example - is just the order of limits, and it is generally recommended to increase the value by 1 for each. There are known exceptions in vanilla files.

As a final note, this **does** work to randomise BGM, **assuming** that you're attaching it to animations or VFX, because it doesn't get counted as "true" BGM. You *cannot* randomise actual BGM. Additionally, you must use a file that already has the audio set up; trying to add new ones will fail to be recognised for the *Type*. Unsure why or if this is still an issue.

GroupOrder

Not as advertised. Test at your own risk. I enabled this on a set of tracks that were originally *GroupRandom* and made the game freeze. :)

Atomosgear

Directly relates to its flag with the same name and sets the file as capable of changing according to player count. See the *Atomosgear* flag section above for more information.

ConditionalJump

This is another I haven't seen used. Quite possible that it can act as a switch. Impossible to know for certain without the extra provided information in the *Tracks* subtab to guide us.

Empty

Reserved for SCDs that have no sound. They instead carry their own arguments elsewhere. Those tend to be *IsLittleEndian*, *UseDistanceFilters*, and *IsWooferOnly* in *Layout*; and *[NoPlay]* in *Attributes*.

MidiMusic

To hazard a guess, this is called for MIDI files. Not that I've seen them, but you know.

Local Number

This is essentially a way to uniquely identify a sound within a related set of sounds, but it also determines if a sound (with the same local number) should duck on subsequent overlapping plays. (Thanks, MaxValparaiso, for mentioning this.)

Thus, you could alter the local number to others if you wanted to have the same sound play multiple times with no clips, but this is better achieved by modifying *First Condition* under the *Attributes* tab, as a changed *Local Number* may interfere with other unmodified sounds as a result. See Appendix E for more info.

User Id

A big question mark. Never seen this at anything besides **0**.

Play History

Unknown. Mostly **0**, but can also be **4** and **-1**.

Example of **-1**:

sound/vfx/lbk/SE_VFX_LBK_Squad_Lv2_NPCHatsudo_c.scd

Appendix C: Sounds > Layout

Type

Allows you to set how many emitters (points where sound emits from) are in a given space, determined by what shape it is or direction it goes. All relevant fields will show up below the separator line. For example, if you pick *Surface*, the bottom half of the window will update to show four positional entries with four fields each, and you'll get a series of other determiners.

Unsure if **Flag3D** would need to be checked if it's a polygonal shape.

I have also never seen anything but *Point* being used for *Type*, and getting the other types to work on general sounds can be difficult (or they'll crash you with no warning).

Some files types, like SGB, can externally modify the *Type*, if need be. For example, *bg/ex4/05_zon_z5/shared/for_vfx/sgvf_z5r1_b2474.sgb* will apply a *Type* of *Surface*.

If anyone finds something besides Point for *Type* in SCDs, please send me a message. I've been looking forever and I'd like to know if it actually exists internally.

Null

As it says, this will nullify the sound. This is more or less equivalent to either having no *Audio* entry or a *Type* of *Invalid* in the *Parameters* tab.

Ambient

Volume, Pitch, Reverb FAC, Direct Volume 1, Direct Volume 2

Direction

Volume, Pitch, Reverb FAC, Direction, Rotation Speed

Point

This is the default *Type*. **It's highly recommended to just use this one**, unless you're really into testing and don't mind crashing unexpectedly.

PointDir

Point, but with some directional arguments.

Direction, Range X/Y, Fixed Direction

Line

Seems to imply that it'd spawn a sound across a defined line.

Start/End Position, Direction

Polyline

Holy positions, Batman! Yes, you see that right - you can have up to **16** XYZW (quaternion) positions set. You can also set a *Vertex Count* and a *Direction*. If you're new to the concept of a polyline, it's essentially just a sequence of interconnected lines that can give the illusion of a curve, if so desired.

Surface

In order for this sound to work, the target object must be rotating. If spawned on your character, then your camera must be rotating beyond a certain speed threshold.

The *Type* itself will allow specification of *Position 1-4* (XYZW at corners?), a *Sub-Sound Type*, an *Is Reverb Object* checkbox, and *Rotation Speed*. Not much of this has been tested on a relevant object.

BoardObstruction

Some sort of sound to be applied as an obstruction, perhaps?

Position 1-4 (likely to be corners again), *Obstacle FAC, HiCut FAC* (which you need to enable with the checkbox), *Open/Close Time*

BoxObstruction

Some sort of sound to be applied as an obstruction, perhaps?

Position 1-4 (likely to be corners again), *Height, Obstacle FAC, HiCut FAC* (which you need to enable with the checkbox), *Fade Range, Open/Close Time*

PolylineObstruction

Some sort of sound to be applied as an obstruction, perhaps?

Position 0-15, Height, Obstacle FAC, HiCut FAC (which you need to enable with the checkbox), *Vertex Count, Width, Fade Range, Open/Close Time*

Polygon

Suggests that the sound is set in a polygonal shape (with *Flag3D*?).

Standard settings, plus *Sub-Sound Type, Is Reverb Object, Vertex Count, Rotation Speed, Position 0-31* (any icosidodecagon fans out there?)

BoxExtController

Unsure. Doesn't come with any additional settings. Doesn't play the sound on your character.

LineExtController

Unsure.

Start/End Position, Lower Limit, Function Number, Calculation Type

PolygonObstruction

Some sort of sound to be applied as an obstruction, perhaps?

32 *Position* settings, *Obstacle FAC, HiCut FAC* (which you need to enable with the checkbox), *Vertex Count, Open/Close Time*

Version

A question mark, as usual. I'm guessing it's a bit like if SE made several versions/revisions of a particular item and chose to use that one in particular. Anyway, it's always **1** here.

Used Fixed Direction

I guess this makes it so how the sound is perceived doesn't change depending on camera/character direction related to the source.

Unbounded Distance

This is omething I'm not sure on yet. Maybe it allows the sound to play at any distance on the X and Y axis. Possibly related to *Bottom/Top Infinity*.

First Inactive

Always checked. Literally always. If you uncheck this, anything set in *Height* will be read, otherwise it's ignored.

Bottom Infinity/Top Infinity

Appears rather like your sound shouldn't be bound to a range on the Z axis. Haven't really tested.

Flag3D

It's here. It exists. Why is it here? *shrugs into infinity*

Point Expansion

Seems to ignore *Max Range* and *Min Range* when checked. Plays at about 50% of total volume if *First Inactive* is unchecked. Found in some cutscene voice files.

Example:

cut/ex3/sound/VOICEM/VOICEMAN_05002/vo_VOICEMAN_05002_001800_m_en.scd

Is Little Endian

Defines if your file has the little-endian encoding, which, from what I gather, makes byte positions differ in that it stores them in reverse, as opposed to big-endian. It shouldn't really be applicable for most imported music that I know of, and I haven't had something be unreadable in that regard, but it's there if you need it. The only place I've seen it thus far is in cutscene audio.

Group Number, Local Id, and Bank Id

These tend to be utilised with certain ambient sounds. TBD on what these are drawing from. I'd also love an example file, thank you.

Is MaxRange Interior

This could be self-explanatory, but it's also never used. One day.

Use Distance Filters

Example:

sound/vfx/monster6/SE_Vfx_Monster_ETC_Eureka_Ether_c.scd

Use Dir First Pos

This will use first direction position as the sound's origin. It can allow a sound to play if it's indoors while the camera is outside.

A sound would normally mute if your camera were set like this outside of a door while your character was inside, so *Use Dir First Pos* would counteract that behaviour.

Example:

cut/ex3/sound/VOICEM/VOICEMAN_05002/vo_VOICEMAN_05002_001800_m_en.scd

Is Woofer Only

I've mainly seen this with things like the aura pulsing around Nidhogg's eye and the flying mount whooshes. This flag also exists in general cutscene audio and some *Empty Types*. My best guess is it goes hand-in-hand with sound modulation and altering only the lower frequencies of a given audio track.

Is Fixed Volume

This will play the sound at a flat volume up to *Min Range*, at which it will begin fading. That volume is determined by what it would have been at *Min Range*.

Is Ignore Obstruction

This probably relates to, well, if the source of the sound shouldn't be obstructed by things like walls or solid objects, but I also haven't gotten this to work in that manner. It doesn't exist in vanilla files either, as far as I know, which is always so fun to see when you want to know how it works.

Is First Fixed Direction

Unknown.

Is Local Fixed Direction

Unknown.

Reverb Type

Untested - need an example.

AB Group Number

Unknown - need an example.

Volume

While there are fifty billion *Volume* fields, the one here has four values you can input, though I'm still trying to figure this one out.

The first value is the only one that seems to make a difference so far and appears to function off of another set value.

This *Volume* as a whole could be related to surround sound or Immerse audio, of which I have neither.

Position

Unknown and somewhat untested. On every sound I've seen, it's always **0.0, 0.0, 0.0, 1.0**. Could be related to surround sound or Immerse audio.

Max Range

This is how far you can be before the sound will start fading out. Most player skills have this set at **10.000**, which is about the maximum manual camera distance you can set. Orchestrion BGM has this at **3.000**, which is the maximum manual zoom-in distance (first-person view). This may also

depend on the *Listening Position* setting under the game's System Configuration - the default position is **10**. I have not run a test yet on if this actually makes a difference.

Min Range

Appears to determine how close the camera(?) (listening position) has to be to the sound's emitter in order to hear it.

Note how this differs from *Max Range*: *Min* will **mute** the sound at *x* units, while *Max* will begin to **fade** the sound at *y* units. The reason for this is the value set in *Lower Limit*. Additionally, the variance between *Max* and *Min* is calculated as a percentage of volume fade. Ergo, if they're equal, you'll get a flat volume at the value, and moving your camera even a tiny bit beyond that will immediately mute it.

If *Min* happens to be less than *Max*, you won't crash, but it does appear to overwrite your *Max* value.

Setting *Min* to anything lower than **2.000** renders the sound inaudible on your own character, even in first-person mode.

Height

Defaulted to **10.000** for the first field and **0.000** for the second. This only seems to make a difference if *First Inactive* is disabled.

The first field affects the vertical (and somewhat horizontal) distance that a sound can be heard. Second field is still unknown. To clarify, if *First Inactive* plays the sound like it's a sphere, then unchecking it turns it into a tall ring. Note that this will heavily distort the affected sound in first-person view when you shift the camera around.

Range Volume

Appears to work as a subset of *Max Range*, affecting the volume percentage only if it's beyond that *Max* value.

While *Lower Limit* is vertical for *Height*, *Range Volume* is horizontal.

Volume

This one with a single field is additive with any other volume changes. A total volume over **1.000** will be negated.

Pitch

Modifies both pitch and the rate at which the sound plays. Using this in conjunction with *Pitch* in *Tracks* will make a rough average of the two.

Do not set this to 0. You will crash.

Center FAC

Determines the total increase in volume from *Min Range*, with a max of **1.000**. Usually turns out to be quieter at max than the sound's actual volume. If set to **0.000**, this setting is ignored.

If you're curious about what FAC stands for or what it's capable of, you can check this page out, as well as this page. They're more focused on guitar pedals in the explanations, but generally, FAC applies a high/low-pass filter, traditionally with capacitors.

Interior FAC

This is most often set to **0.000**, but Orchestrion BGM uses **0.200**. Can sometimes increase audio volume, but not always. Perhaps it's a way to adjust for what something might sound like in a closed space.

Direction

Most SCDs that make use of this have it at **1.571**. This is measured in radians (yay, radians!); the degree equivalent would be 90.

An example of *Direction* can be found in *sound/vfx/monster4/SE_Vfx_Monster_ETC_OverpowerOld_c.scd* with **4.7123890**, or 270°.

Not that I know *how* angle makes a discernible difference, but it seems tied to spatial and/or surround sound. A value of **0.000** will be standard (no pan), while **4.712** will be somewhere off to the side (left?) and **1.571** is slightly closer to the centre (right?). I suppose it would help to have an actual surround setup, or even the spatial audio pack to test, but I ain't payin' for either of those just for the sake of testing.

Near Fade Start / Near Fade End

This seems to be like setting a separate fade distance within the confines of *Min Range*. The *Near Fade Start* value additionally has a distance limiter attached to it, meaning any distance *lower* than what's defined will *mute* audio. This overwrites *Max Range*. Setting both *Near Fade Start* and *Near Fade End* to **0.000** will disable this behaviour.

Far Delay FAC

Unknown.

Use EnvFilter Depth

Unknown.

Is FireWorks

This isn't used on actual fireworks VFX. Appears to do something similar to *Extra Desc* by damping some frequencies.

Is Reverb Object

Unknown.

Is Whiz Generate

Unknown -need an example.

Lower Limit

This is the lowest the volume is allowed to go at distances beyond *Min Range* or *Height*. Usually, you want to keep this at **0.000**.

If you set this above **0.000** (with the max allowable value being **1.000**), your sound will (probably) play *anywhere beyond Min Range*, so you want to be careful about doing that.

Fade In Time / Fade Out Time

This really only makes a difference when attached to BGM, I believe. Default is **2000** (milliseconds). **0** is also seen in older files. Doesn't seem to be any variance otherwise.

Convergence FAC

Currently unknown. (Convergence point?)

Appendix D: Tracks

Same as the other tabs, you can add a new track or new entries within a track, or delete things entirely. Unfortunately, you can't add new ones in between at this point in time, so if you want to preserve the runtime order, you'll either have to remember what was there and start from scratch, remember everything that you replaced so you can add it in underneath, or build what you want in a new track and add in those fields where you need them before deleting the original.

There are far more entries available than I have listed, as they either haven't been tested or aren't seen in any files so far.

ADSR

This stands for "attack, delay, sustain, release". (See also here.) There's probably some nuance to how this affects the game's sound, but you can think of this as a way to circumvent the 1.000 volume limit imposed by the game if you set its value between 1.000 and 2.000. Or you can try **Velocity**.

AutoADSREnvelope

ADSR, but with full control over its options.

ChannelVolumeZeroOne

This can also show up with values occasionally - see *sound/battle/etc/SE_Fly_Kinsetu_Whoosh.scd* in Track 5 for an example, or *sound/strm/GAYA_LestArea_01.scd*. I'll look more into it. Note that you can't actually add any fields in for this, so you need it supplied for you. Has alternating fields of **Version**, **Zero One**, and **Value**.

Config

Contains subtypes of **IntervalType** and **IntervalTypeFloat**. Untested.

CutOff

This seems mostly a way to cut a sound short upon user input or perhaps on cutscene start/end. It exists, for example, on one of E9's Cloud of Darkness' skills (*sound/vfx/monster7/SE_Vfx_Monster_m0684_sp16_c.scd*), cutscene voice lines, and on some ambient sounds.

End

Used at the end of all items to mark the end of it. How unexpected.

EndForLoop

Equally a little mysterious. The name suggests it could end a "for" loop clause, but none of the available fields establish that loop.

Expression

Works somewhat like randomised volume. Haven't tested much on this.

ExternalAudio

Requires a value for its *Bank Number*. For a file like *bgcommon/sound/sea/sea_bg_sea_d_rain_coast_spot.scd*, there are multiple tracks that have the same *Bank Number* of 5011. There's also 5001 in another ambient spot file. Difficult to say where these bank indices exist. 5011 will replace your sound with crickets, though, so that's something.

Filter

Contains several subtypes like **Bypass**, **LowPass**, **HighPass**, and **LowShelving**. A frequency, InvQ (inverse quantile function value), and gain can also be set. Untested.

FrPanning

This doesn't appear to change anything in my test file. Will try others later.

Interval

This field has quite a lot to it. The first instance you use it outside of **KeyOn** determines when a sound will play after it's been called. Any use of **Interval** outside the key flags after the first shows the duration that any potential modulations will occur.

Interval inside of **KeyOn** and **KeyOff** tells the game how long to play the sound uninterrupted by any previous or following modulations.

Interval along with the **LoopStart** and **LoopEnd** flags, preceded by at least a **KeyOn**, as explained before, works to establish loop values. I'll go over it again in the corresponding section.

Generally, though, you want the sum of **Interval** values (**RandomWait** is included in the total as well) to equal your sound's length so that it doesn't get cut short.

Jump

Contains **Condition** with of LZE and LNZ (and [UNKNOWN - 0]). Also an Offset field. Unknown and untested. (Why do those acronyms seem familiar?)

MidiExternalAudio

Contains a **Bank Number** field. Probably needs to be used in conjunction with **MidiKeyOn**.

MidiKeyOn

Contains dual float **Value** fields. Likely disabled with **MidiEnd**. Looks to be a depreciated **MidiKeyOnOld** item as well. Untested.

Modulation

This opens up more options for base modulations of volume, pitch, pan, and FRPan. You set the carrier type, what modulator it'll use to change the affected effect, and the curve of the waveform. Depth and rate are, I assume, to do with bit depth and bitrate.

Note: **Do not** set carrier to *None* and play the sound while you have arguments active below that. Your game will crash.

ModulationDepthFade / ModulationSpeedFade

Contains **Carrier**, **Depth**, and **Fade Time**. Untested.

ModulationOff

You may think of this as a safety net. Whichever modulation you don't plan to use, place that as its carrier. Don't double up and **don't** set it to *None* or you will *absolutely* crash and burn to the desktop as I just did.

Panning

Defines speaker/headphone panning. Broadly, set this between **0** and **-2** if you want left pans, and **0** and **2** for right pans. You can still get left panning with other negative values, but there's not much reason to make the numbers larger.

Time is how long the panning takes to occur.

Pitch

This works the same way as in *Layouts*, modifying the overall pitch and speed, but with the added benefit of the *Time* field, which determines the speed (in milliseconds) the pitch will change to match the value above. **Do not** set the pitch value to **0** or you will crash.

This Pitch field and the one in *Layouts* may not work together. In addition, it works independently of any **Interval** or **RandomWait** value set in the **KeyOn**/**KeyOff** section, meaning even if your sound is slowed down into oblivion, the **Interval** and **RandomWait** values will run as if the sound played at normal speed.

If you wish to modify pitch *without* changing speed, it looks like the method SE uses is to make a **RandomPitch** flag and set both to the same value.

PitchBend

This appears to function exactly the same as **RandomPitch**.

Be sure *Upper* is greater than *Lower* **and** is neither 0.000 nor negative, or you will crash.

PlayInnerSound

Contains a **Bank Number** and **Sound Index**. The *Bank Number* will draw from another sound in the same bank. So, if we have the path *sound/battle/mon/10000.scd*, the bank is (essentially) ***/mon/** and the *Bank Number* is **10000**, and then it'll look for the correct *Sound Index* specified in that *Bank Number*. While this means you could refer outside of your current SCD, the downside is that the bank is more or less predetermined, as is likely which bank numbers are allowed.

How SE chooses to use *PlayInnerSound* is to reference the same file, but have the referenced *Sound Index* contain a *Type* with randomised audio so that you essentially get randomised sounds in a randomised sound.

Examples of *PlayInnerSound*:

sound/battle/mon/13135.scd (lunar whale mount): Track 3

sound/foot/dev/6991.scd: Track 1-11

PortamentoOn

Portamento itself refers to pitch shifting from one note to another. Contains **Portamento Time** and **Pitch** fields. Untested, but likely extremely similar to the various other Pitch items. **PortamentoOff** exists to disable the effect.

RandomPan

See the above **Panning** for what numbers do what. Accepts negative values.

RandomPitch

Functions like **PitchBend**? *Upper* is your maximum allowable pitch change, while *Lower* is the minimum.

Be sure *Upper* is greater than *Lower* **and** is neither 0.000 nor negative, or you will crash.

RandomVolume

Randomises volume within the bounds of other volume settings. *Upper* for highest value, *Lower* for lowest value.

I'm sure I don't have to mention this, but I will anyway - **do not** set either of these below **0** or, as the infamous Mizzteq would say, you will die. *RandomPitch*, same deal.

RandomWait

Applies a randomised delay in a set interval. *Value 1* is the minimum, while *Value 2* is the maximum. You might think of this as a RandomInterval flag, if it helps.

ReleaseRate

Sort of like a fade after the audio is set to finish. Might be used together with another effect, but so far I haven't gotten a response on standard audio. See also ADSR. A value of **50** can be seen in cutscene voice lines.

Sweep

Sweep ≈ **Pitch**, I guess? More info here.

Transpose

Transpose = **RandomPitch** = **PitchBend**?

Unknown64

As said on the tin, it's just a big unknown right now. Contains **Version** and **Unknown 1**. Untested.

Velocity

This is essentially how fast a sound travels in a given space and medium. This affects overall volume and has a higher allowable threshold than **ADSR**. Note that if you stick it on long sounds or music, it will also affect its rate of decay (i.e. its fade time).

Version

Version changes with each SCD. Certain types appear to have the same number, if I recall correctly, but this doesn't match with the version number in ***Sounds > Sound # > Extra*** or of that in *Attributes*. This remains a large unknown.

In the case of *sound/vfx/ability/SE_VFX_Rol_Range_SpeedUp_c.scd* (Peloton), setting this value any higher than **11** or any lower than **0** will mute the sound entirely.

Volume

Changes volume, perhaps percentage-based in conjunction with the other volumes elsewhere. This field almost always exists and has a base/max value of **1.000**. It has its use if you have a starting volume set in the main *Sounds* tab and want to change this after a certain amount of time has passed (e.g. after an **Interval** flag).

You can additionally have more than one in a single track if you want to adjust volume over a specific duration:

The first way is to have it in succession - one *Volume* with 0.000 and *Time* 0 to start, and then a second *Volume* with 1.000 and *Time* # to end.

Another way is to have them separated by Intervals - a similar result but with this additional option to wait for a set duration. You could also do a RandomWait instead of Interval.

VolumeZeroOne

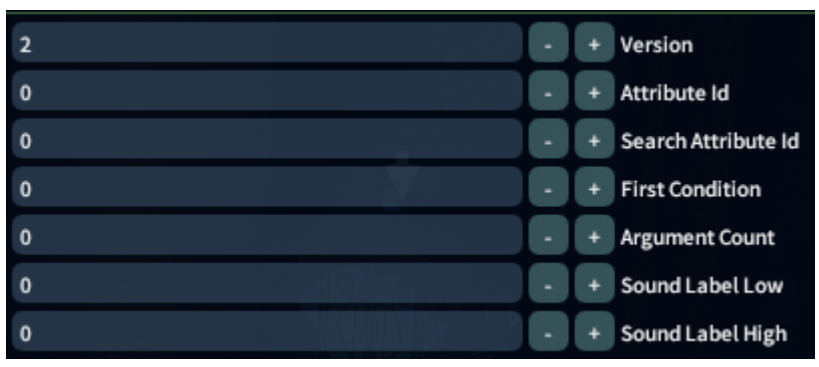
Seems similar to *ChannelVolumeZeroOne* in a lot of respects. Unknown what this does so far.

Example: *bgcommon/sound/bg_ex01/bg_ex01_randomwind_grass_a01.scd*

Appendix E: Attributes

You can't add attributes, unlike the other tabs. Some files do have additional attributes - see *sound/voice/Vo_Battle/Vo_NPC27_Battle_ja.scd* (Y'shtola's combat voices) as an example.

Attribute



Version

This changes depending on what kind of sound it is. Although it doesn't make a difference in casual testing, I'm sure it does something specific. Most have this set to **2**.

Attribute ID / Search Attribute ID

Attribute ID works as an identifier for that particular SCD, more so than something like *Local Number*. *Search Attribute ID* can then reference *Attribute ID* and apply certain arguments to it, done via the *Result* and *Extend* tabs.

Example: If one SCD specifies a *Search Attribute ID* of **4** with a *Result Target Command* of **Stop** in the *Result* tab, that means it's looking for an SCD with *Attribute ID* of 4 in order to tell that SCD to stop playing.

First Condition

This number generally seems to match whatever *First Condition* under *Extend #* is, meaning **0** for None/Unknown and **4** for Equal. Exceptions exist, and it tends to have its own function, I guess, so this will need to be investigated.

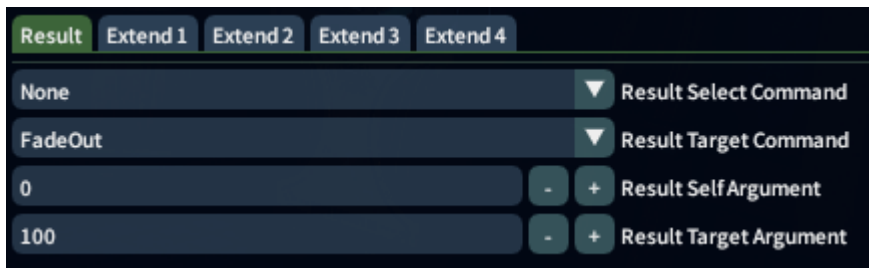
Argument Count

Refers to how many *Extend #* tabs are being used.

Sound Label Low / Sound Label High

TBD. Most will have this at **0** / **0**, but *SE_VFX_common* has it at 2011039 / 2011039 in *Attribute 3*.

Result



The *Result* tab on its own will always apply to the given *Search Attribute ID*. When seen in conjunction with *Extend #* fields, it'll only apply when *Extend #* is being used.

Result Select Command

Unknown.

Result Target Command

Contains a series of commands that the SCD will follow upon being called. If **None**, then nothing special happens.

Stop / PriorityStop / OldStop / LowExternalIdOldStop / MinVolumeStop seem to be generally equivalent in test cases, functioning to stop the affected SCD. See below for more *Result* info as it pertains to *Extend* tabs.

Extend

First Condition

Usually set to *Equal*, which lets a sound overlap on repeated use (if applicable) and is dependent on the value set in *Number of Conditions*, as well as what's set in *Result Target Command*. To provide an example, the default for *Number of Conditions* is **1**, meaning only one other sound is allowed to play simultaneously on the skill in question. This includes from all sources: VFX, TMB, and PAP sounds. If you wish to have a certain sound play fully, you want to increase this value to be one higher than the maximum number of sounds that will play in that skill at a given moment. For example, should you be using a skill with a number of charges that can be pressed in quick succession - say, Onslaught with three charges - and you have five sounds per use of it, you'd probably want to set the *Number of Conditions* to **16**. That assumes all the sounds will last across all three uses of the skill.

If *First Condition* is anything besides *Equal* or *Bank*, the sound will overwrite itself by default, though this is possibly as a result of incorrect settings during tests.

External ID First can be found on any of the *sound/strm/gaya/** files. It seems to suggest that it'll check for the *Search Attribute ID* in an external SCD before trying to apply to itself.

Second Condition

Pulls a series of Lua-based conditions and variables directly from game code in order to connect arguments. While what everything does as a whole is still beyond my ken at this point, I can at least expound on some of the abbreviations: *GT* is greater than, *LT* is less than, *LE* is less than or equal to, *EQ* is equal to, and *NE* is not equal to.

Join Type

Seems to work in tandem with *Second Condition* (and *First Condition*?), although how it does it still a mystery to me.

Example of *Join Type Or* (with *Attribute ID* of **51**): *sound/battle/etc/SE_Bt_Etc_AOZ_Score_Start_loop.scd*

Number of Conditions

Defines the number of conditions that will be activated per use of the *Extend #*. Works in tandem with *First Condition* and the *Result* subsection below it.

Self Argument

This is generally just **3** if there's a Condition specified above it, or **0** if there's nothing (or sometimes if there's something, I guess?). I don't know what this does specifically. Some files do use other numbers and it can be sequential if there is more than one *Extend #*.

Result

This subsection in the *Extend #* tab still needs to be experimented with. I assume from its nomenclature that it'll activate based on the condition or argument specified. So, something like *Result Target Command* of *FadeOut* with a *Result Target Argument* of **20** will mean that if the conditions are fulfilled, the sound will fade out over a duration of 20 frames. Perhaps this means that if a normal battle sound has *MinVolumeStop* and you change it to *None*, it'll let the sound be uninterrupted. Maybe. Again, it'll need to be tested.

OldStop example: *sound/battle/etc/SE_Bt_Etc_MG_7thPazzle_AutoSet.scd* (for the puzzle minigame in the inn)

The following settings will disallow immediate repetition and mute its sound temporarily on reuse (i.e. until the original sound's length has expired):

Equal / EQ / Or / 2 / 3 / 0 / ChgVolume / 0 / 3 / 0

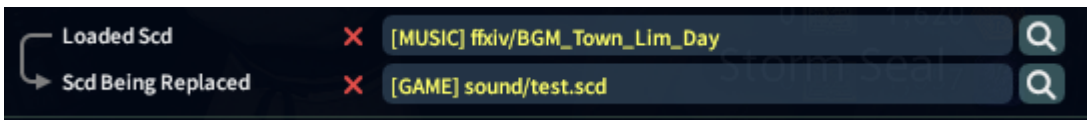
Example file for multiple *Extend #* and *ChgVolume* attributes: *bgcommon/sound/spot_wtr/spot_wtr_Sticky_Mud_On5_c.scd*

Appendix F: Using custom paths

Given the massive hullabaloo over SE's internal renaming of textures and some VFX files in patch 6.5, I believe it's becoming far more important to adopt the idea of using fully custom paths (where possible) in order to future-proof your edits. Thus, I would recommend looking this section over and learning how it works. The below explanation applies to sounds - as this *is* a sound-based document - but it can be applied to all manner of file types.

For our example, let's say you're using Limsa's day theme as a template for your custom sound. You have it set in *Loaded Scd*. You've already changed the internal sound to a custom OGG or WAV. But you don't want to overwrite Limsa's sound or any other vanilla sound with your custom sound here. You'd like to keep those the same. This is where custom paths come in handy.

In the *Scd Being Replaced* field, directly type in the path you'd like to use and press enter.



But what should the path be? Well, there are a couple ground rules:

1. It needs to be contained in at least one folder. In other words, it can't just be, for example, *mysound.scd*. It must be, at the very least, ***sound/****mysound.scd*. You can have as many folders as you'd like for the path - like *sound/sound1/sound2/sound3/sound4/mysound.scd* - but the less there are, the less the hard drive has to work to access it.

Note: You should not be physically creating these folders unless your mod is imported into Penumbra already and you need to edit it to include a new folder. The editor takes care of any and all folder creation in its export process, if you choose to go that route.

2. One of these folders in the path **must** start with something the game recognises as valid according to its database. This is to prevent a potential crash.

A short list of accepted folder names:

vfx/ music/ sound/ chara/ bgcommon/

Example paths with valid structures:

sound/test.scd

chara/hello.scd

chara/vfx/sound/happyisayuppieword.scd

sound/gator/hemlo.scd

3. When you're establishing a custom path, this is *not* its full local path. In other words, it **should not be** something like *C:\Penumbra mods\Fortnite Dance Mod\sound\fortnite.scd*. This will crash you immediately if you try to play it. It should instead be the last part of that local path, the part that's *inside* the folder of your mod: **sound/fortnite.scd**. The direction of the slashes probably doesn't matter, but it's best to have it as **/** to be safe.

4. Your custom path **will not cross-reference with other mods**. If you try to access a path from a different mod, it won't work. The custom paths may *look* the same, but they are functionally entirely different. However, that does mean you can have the same custom path for every single mod, if you so choose.

5. This one is semi-optional, but best to consider for compatibility across mod loaders. You ideally want to keep your custom path in lowercase and not mixed/camel case; and avoid special characters. You might also possibly refrain from using underscores and dashes, although I imagine it shouldn't cause too many problems these days. Additionally, accented or non-English letters are known to be problematic. Chinese/Japanese script has mixed success, likely depending on whatever font your game client supports, but it's better to avoid in case.

Examples of path structures to potentially avoid:

sound/ThisIsMySound.scd

Sound/VFX/Custom/helloThere.scd

chara/jedi_lightsaber_sound-new16.scd

music/音楽/じゃがいもですが.scd

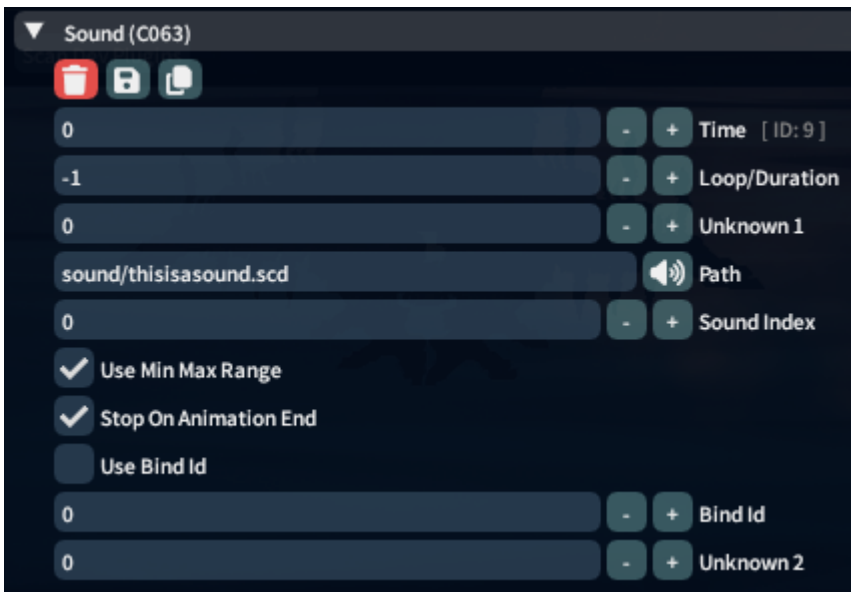
Examples of special character path structures to avoid:

sound/my$sound^new.sound#@.scd

sound/weiß/unpetitgâteaubrûlé.scd

Once you've established your custom path in the editor, now you can use it in other parts of the editor.

PAP/TMB example:



VFX example:

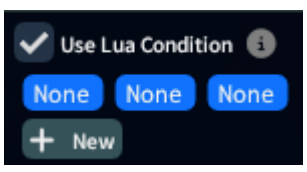
Appendix G: Using Lua conditions

Lua conditions are a way for FF14 to interpret how to deal with a track in a TMB or PAP file; these conditions are not set in the SCD itself. The most common condition is one that will let you limit a track to the person that's currently using a given TMB/PAP. In other words, you can make it so something like a sound or VFX will only show on that player's screen. FF14 tends to place this condition on VFX like screen shakes or radial blurs, although it can show up on some mount skill sounds as well (like the Moon-Hopper one).

While there are several known conditions besides that (of which you can read all about those in the TMB/PAP document), we'll focus on just the user-based one for now - **Is Player Character == 1**. To enable this condition, you'll click the **Use Lua Condition** checkbox in the track you wish to user-limit:

Side note: If you hover over the **i** next to the checkbox, it'll tell you where to find what's being used for Lua conditions.

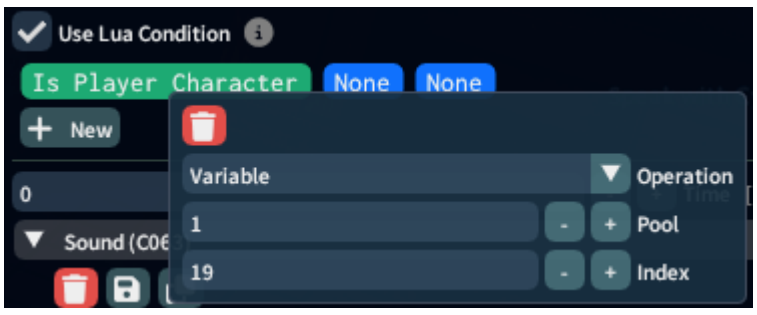
Click the **+ New** button that appeared below it - **three times** if you only want to use it on one track, or **five times** if you plan to do this on multiple tracks in the same TMB/PAP. You'll now have this:



OR

I'll try to guide you through this as simply as possible, since it can get rather technical.

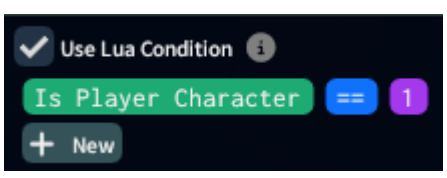
Click on the first **None** entry. If you have **five None**, click the second entry. In the window that popped up, click on the dropdown next to **Operation** and pick the list item that says **Variable**. Then type in **1** in the **Pool** field, and then **17** in the **Index** field. You should have something like this, where what was once **None** is now green and reads **Is Player Character**:



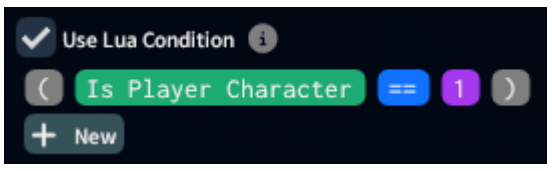
> Side note: These operation colours may differ if you've customised your VFXEditor interface in *File > Settings > Tmb*. If they don't, well, now you know how to make them different.

For the second **None** (or third if you have five **None**), make its Operation **Equal**. It will show up as **==**.

For the third **None** (or fourth if you have five **None**), make its Operation **Int_Value** and its Value **1**. You should have this if you've done it right and only had three **None** to start with:



If you had five **None**, let's deal with the bookends. For the first **None**, make its Operation **Open_Parens** and its Value **1**. For the last **None**, make its Operation **Close_Parens** and its Value **1**. You'll have this:



That's all there is to establishing a track to be user-only. If you wish for something to be everyone but the user, you can try making it **Is Player Character == 0**. Although I have not taken the time yet to verify if this works or how it affects things, theoretically it should be as expected.

(Incomplete) list of SCDs and what they affect

Below, I've gone into small detail about possible .scd paths within FF14 index files.

For a more comprehensive list (which is mostly obsolete now, but meh), check here. If you'd rather have the full dump of the known database list without descriptions, including files beyond .scds, check perchbird's RL2 site. They have options for both the current game version and the entire database, in .txt and .csv formats, and the ability to use a .gz in conjunction with coding languages.

I've also been given permission to share Vulonsulin's spreadsheet collection, which contains quite a few SFX/voice lines that I haven't had a chance to document yet - as well as a large assortment of VFX and animation paths, and a raw path list dump. This may be a bit outdated.

general .scd path descriptions (within XIV Explorer):

Note that a few .scds (or files and folders in general) are listed as *~(random string)* (e.g. *~2defad7d*). This is due to a missing hash - i.e. it hasn't been found in-game yet and subsequently recorded with Reslogger2, so the path hasn't been established. It's recommended to download the latest paths, which you should be able to do easily in goats' FFXIV Explorer fork (you will need JRE/JDK 11 to use this program). If you're still unable to see what their actual name is in XIV Explorer, they do show up in VFXEditor's ***Tools > Resources*** or ResLogger2, provided you run into them within the game.

Also, if you're lacking a million brain cells like I've been for the past ***year***, be sure you're looking in more than just the base *ffxiv* index folder for files beyond ARR. Yes, you can, in fact, go up in the folder tree to find the other expansions' index files.

ffxiv (ARR and shared) indices

index 010000

bgcommon/sound/abr/ - environmental sounds (e.g. fountains, airships, rivers, windmills), as well as a couple instance-specific ones. seems mostly tied to Azys Lla or Allagan areas

bgcommon/sound/bg_wth01/ - Shadowbringers and Endwalker content

bgcommon/sound/dra/ - Dravanian areas and dungeons

bgcommon/sound/emp/ - Garlean empire-based

bgcommon/sound/est/ - Othard/Hingashi/Far East areas and dungeons

bgcommon/sound/fst/ - Black Shroud sounds. some also extend to any other zone (e.g. aetherytes and instance markers). might think of this as a catch-all for reused assets

bgcommon/sound/gyr/ - contains three "melt" sounds

bgcommon/sound/hou/ - housing item sounds

bgcommon/sound/lak/ - lake-like areas

bgcommon/sound/mgc_blk/ - NPC BLM stuff?

bgcommon/sound/mgc_chm/ - one found placement is the radar in Base Omicron. difficult to say what the shorthand is, then. might be "chemical", but it doesn't match the aforementioned example

bgcommon/sound/mgc_oth/ - anything that doesn't fit cleanly in the other *mgc_* folders

bgcommon/sound/mgc_set/ - unsure

bgcommon/sound/mgc_vid/ - void-based SFX. Apparently, there are enough of them to have a separate folder

bgcommon/sound/mgc_wht/ - NPC WHM stuff?

bgcommon/sound/ocn/ - ocean-based SFX

bgcommon/sound/roc/ - includes doors, caves, snow, rivers, and Garuda-related sounds. generally, Coerthas or mountainous/cavernous areas

bgcommon/sound/sea/ - sea-based sounds, but also has things related to jungles, coasts, towns, kobolds, doors, cooking, and rivers. La Noscea is the main affected area, but I've seen some places like Radz-at-Han and general locations with water also use these files

bgcommon/sound/spot_crt/ - creature sounds. also includes some moving objects, which is, in a game-design sense, sort of like an ambient creature, I guess?

bgcommon/sound/spot_dor/ - door sounds

bgcommon/sound/spot_elc/ - electricity-based sounds

bgcommon/sound/spot_etc/ - sounds that don't fit cleanly in other *spot_* folders. includes fans, chains, windmills, glass breaking, creaking, and impacts

bgcommon/sound/spot_evt/ - sounds for specific events (e.g. some Gold Saucer and seasonal)

bgcommon/sound/spot_fir/ - fire-based sounds

bgcommon/sound/spot_mtl/ - metal and machine-based sounds

bgcommon/sound/spot_plt/ - plant and wood-based sounds

bgcommon/sound/spot_stn/ - rock, sand, and stone-based sounds

bgcommon/sound/spot_wnd/ - wind, gas, and steam-based sounds

bgcommon/sound/spot_wtr/ - water and aqueous sounds

bgcommon/sound/wil/ - seems to be anything Thanalan. extends to Gyr Abania and desert areas

bgcommon/sound/zon/ - mostly alliance-based sounds, sometimes broadly used areas (jail ambience and airship ambience for cutscenes)

index 030000

*cut/ffxiv/sound/** - ARR cutscene sounds and dialogue

index 0c0000

music/ffxiv/ - general zone music, mount music, leves, fishing voyage, events, seasonal events, Gold Saucer, ARR dungeons and trials

music/ffxiv/movie/ - music for dedicated CGI cutscenes and movies

music/ffxiv/orchestrion/ - just like it says. orchestrion songs

index 070000

sound/battle/wep/ - general weapon sounds for any job and also any NPC

sound/battle/live/ - DoH/DoL skills

sound/battle/mon/ - minion/mount/monster/enemy sounds + steps (listed only as numbers)

sound/battle/enpc/ - NPC sounds (encounter NPC)

sound/battle/etc/ - mostly mount sounds, but can include misc. things related to combat

sound/battle/pc/ - player character/NPC sounds

sound/bg_obj/sea - only contains *s_oast_i_aqualium.scd*. I assume it's just the aquarium for housing, but then wouldn't it be with the other housing sounds…?

*sound/cut/** - cutscene stuff

sound/cut/arrangement/ - door sounds

sound/cut/ascian/ - reserved for only two sounds: when ascians warp in and out

sound/cut/battle/ - generic battle hit sounds (for cutscenes)

sound/cut/boss/ - a couple generic boss sounds like the cutscene zoom-in before a fight or when they die in the cutscene after

sound/cut/camera/ - camera whooshing sounds

sound/cut/effect/ - any effects in cutscenes

sound/cut/explosion/ - explosions in cutscenes

sound/cut/hdb/ - Hildibrand-specific sounds

sound/cut/kako/ - sounds for Echo cutscenes

sound/cut/movement/ - any movement-based cutscene sounds like falling, clothes, jumping, putting things on

sound/cut/props/ - general props like books, dishes, machines, and linkshell

sound/cut/roar/ - unsurprisingly, roars

sound/event/ - highly varied sounds (doors, water, dog barking, jingle bells, rumbles) probably tied specifically to seasonal events

sound/foot/dev/ - footsteps, but labelled with numbers, and only from 6994 to 6999

sound/foot/foot/ - footsteps for all terrain types

sound/instruments/ - for BRD performance (several sounds per .scd, each generally an octave apart)

sound/nc/general_ex1/ - sounds specifically for the first expansion, Heavensward. may be some repeats from other folders

sound/score/ - seem to be only test beeps (in .msb format)

sound/strm/ - ambient crowd sounds for highly populated areas. includes a blank *gaya_nosound.scd*

sound/system/ - system sounds for the UI, Gold Saucer, Doman mahjong, Triple Triad. additionally, a few test sounds

sound/vfx/ - contains a single *se_vfx_common.scd* for generic player-based sounds like shield pings, buffs, debuffs, teleport/return, as well as some harp jingle things

sound/vfx/ability/ - any DoW/DoM abilities not specifically listed as a spell or weaponskill. also includes role skills

sound/vfx/etc/ - many, many random sounds for things that don't fit cleanly in other folders. examples are some special emotes gained through items, aetheryte channelling sounds, splashing sounds while flying a mount on water, Bozja/Eureka skills, and the sound an NPC makes when they spawn

sound/vfx/guild/ - aside from *se_vfx_lvup.scd*, which is the swoosh sound when you level up, and some levequest ones, most seem only to be test beeps when played in VFXEditor. Perhaps they're placeholders for unreleased content?

sound/vfx/item/ - item-based sounds, specifically deep dungeon items and seasonal ones

sound/vfx/lbk/ - limit breaks

sound/vfx/live/ - DoH/DoL sounds not tied to specific skills

sound/vfx/magic/ - skills specifically labelled as magic

sound/vfx/monster[#]/ - enemy and combat-object sounds

sound/vfx/weponskill/ - skills (marked as weaponskill) for jobs

sound/vibration/live/ - vibration sounds specifically for FSH line tugs

sound/vibration/preset/ - vibration sounds for specific presets

sound/voice/vo_battle/ - battle voices for each race and for NPCs in all available languages

sound/voice/vo_cm/ - sample voices you hear in character creation in each language

sound/voice/vo_emote/ - very, very large database of emote sounds (listed only as numbers)

sound/voice/vo_line/ - every voiced NPC dialogue line in every language (listed as numbers)

sound/zingle/ - things like quest/raid/FATE start/finish sounds, grand company sounds, leve sounds, PVP sounds. anything that could be classified as a jingle

ex1 (HW) indices

0c0100

music/ex1 - HW-specific BGM

music/ex1/movie - a singular file for the opening HW movie

030100

*cut/ex1/sound/** - HW-specific cutscene sounds

ex2 (SB) indices

0c0200

music/ex2 - SB-specific BGM

music/ex2/movie - singular file for the opening SB movie

030200

*cut/ex2/sound/** - SB-specific cutscene sounds

ex3 (ShB) indices

0c0300

music/ex3 - ShB-specific BGM

music/ex3/movie - four files for language-specific variants of the ShB movie

030300

*cut/ex3/sound/** - ShB-specific cutscene sounds

ex4 (EW) indices

0c0400

music/ex4 - EW-specific BGM

music/ex4/movie - four files for language-specific variants of the EW movie

030400

*cut/ex4/sound/** - EW-specific cutscene sounds

ex5 (DT) indices

0c0400

music/ex5 - DT-specific BGM

music/ex5/movie - four files for language-specific variants of the DT movie

030400

*cut/ex5/sound/** - DT-specific cutscene sounds

SCD research for SFX and BGM (for possibly insane users)

My research has been moved here, as it was getting far too long and beyond the intended depth of this document. I've enabled comments there, in case someone catches something or has an answer to a problem within.

There is also a minor section for FFXVI assets, since I was curious about it.

Troubleshooting: I've got 99 problems and this is one

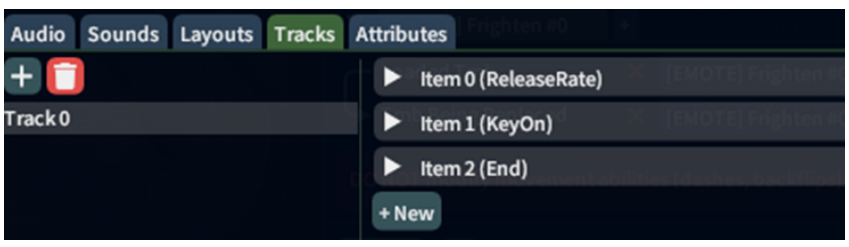
I'm getting strange pitch and speed changes in my converted sound!

This seems to be a very common problem that I get frequent messages about and is caused by one of two things:

1. **You're doing conversions through XIV Explorer.**

   It's great for some things, but definitely not for SCD conversions. Until that gets a proper update (been nearly four years), drop it like it's hot.

   I have no idea what settings it's trying to use or if it's an outdated Ogg2Scd version with a bug, but 80% of the time your output file will be botched somehow. It also does this weird thing where it deletes a bunch of track items that the game would prefer to have:

   And it hardcodes a loop or something. Basically, it's being a problem child. We don't like problem children here.

   Thus, conversions through VFXEditor are *highly* recommended, but Sebastina's Voice Pack Creator is also acceptable. Just note that the latter method will produce an SCD that'll be marked as unsupported in the editor and could do unpredictable things when messed with further (and may also do the same thing as XIV Explorer with metadata). There are a couple other things about VPC that aren't overly friendly when you're working with music (its original intent was for voice swaps, after all), but generally it's fine.

2. **Your converted sound has a frequency (kHz) mismatch.**

   More or less, whatever your sound's sample rate is (like 44.1 kHz or 48 kHz), it's best if the converted file matches. Sometimes this won't make a difference, but other times it'll cause undesired playback issues because the encoder did something it shouldn't have.

   If you're using Audacity, click the three dots (...) next to the track name, hover over *Rate*, and the sample rate your file uses has a checkmark next to it. Or in Windows, you can check this by right clicking your unconverted sound, clicking *Properties*, and going to the *Details* tab in the pop-up window. It'll be the *Audio sample rate* under the *Audio* heading.

   Okay, yes, I may have suggested to upscale to 48 kHz for better looping before because I never ran into issues doing that myself.

   But now I'm suggesting not to do that because it just turned the sound into molasses for someone else.

What happened to WAV for BGM? Nothing plays if I try it!

I don't know when this happened, but the game no longer supports WAV playback specifically for BGM. The bus number itself is still fine, so you're welcome to go ham with stuff like music on emotes or emitters, but a direct replacement over a BGM file will result in silent audio, even if using a vanilla file with WAV already set up. Perhaps this coincided with the introduction of HCA in 7.5. Or maybe there needs to be a specific hook set up in plugins now to make it happen. In any case, you're basically stuck with lossy formats for BGM unless plugins enable support for WAV on the BGM channel.

I imported a WAV file and crashed!

[VFXEditor bug as of 1.8.3.7] Make sure your WAV encoding **is not Microsoft ADPCM** (some programs may have this as **4-bit ADPCM**) **or IMA ADPCM**. If it's **16-bit** or **32-bit**, you're fine. If you still crash, you may also try deleting the *Audio #* that contains the sound you want to replace and then adding a new *Audio #* with the **+** button before importing.

With **Microsoft ADPCM** or **IMA ADPCM**, the resulting SCD will mute *all* sounds for its duration. I don't know why this is (container unable to be read properly by the game? even though MS ADPCM is natively supported???). That means, unfortunately, you need to use one of the large-sized WAV encodings instead. I mean, you *could* go for U-Law (μ-law) or A-Law, but those are a bit stinky in terms of quality, mostly for telecommunication stuff.

What does the editor mean by "! Unsupported"?

Basically, this means the file can't be properly verified. Something or other is non-standard with how it was made, and the editor can't guarantee that it will work flawlessly. This boils down to three common sources for this error: 1) it's an old SCD that contains metadata that's no longer used in some manner; 2) the SCD was processed in an outside program like FFXIV Data Explorer or Voice Pack Creator, where some unexpected data has been removed or hardcoded; or 3) a WAV file has been imported in a recent version of the editor (more or less, since early-mid 2024), is conflicting with new metadata additions under *Audio > Parameters*, and thus has not been coded yet to be verified in that file type.

While usually you can ignore this type of error, sometimes it can cause problems when edited again, akin to when you see "X Parsing Error". Therefore, to ensure maximum compatibility at this time, be sure to work with SCDs only through the editor, and with OGG. (The main person behind VFXEditor is currently on hiatus, so try not to expect a fix for this anytime soon.)

If you have some decent programming know-how, you can additionally revert the relevant .cs files back to before the *Audio > Parameters* information was added (before March 3, 2024 - the Github commit marked as *scd parsing overhaul* is roughly when this started happening, and when samples changed to second values, although note that with a broad sweep reversion, you will lose out on the marker metadata for multi-channel audio). This will at least fix the constant Unsupported error thrown by custom WAV files on re-opening the SCD, and perhaps also fix their loop issues.

My mod music isn't playing in my Penumbra character collection.

You need it in the default collection. It's just Penumbra being Penumbra and has nothing to do with any particular mod. Penumbra used to allow character-specific BGM (which was great for altoholics like me that like themes to match their character), but has since chosen not to because I don't know.

I just created/imported a new mod into Penumbra and its sound isn't playing.

If you've just created or imported a mod that swaps an SCD path that hasn't been swapped anywhere else before that, you may run into a problem where the mod won't properly switch it. For most cases, clicking **Reload Mod** under the *Edit Mod* tab in that Penumbra mod will take care of this. (Make sure you're not currently trying to play this sound while you do this, especially with BGM.) Otherwise, going to a different map or logging to character select and back should allow it to work. If even that's failing you, try a hard restart of the game.

I still can't get my modded music or modded UI sound to play.

For all intents and purposes, music and non-character/non-job-specific SFX mods should be enabled in your default collection. Even if it seems like they might be tied to the interface collection (such as the FF7R Test Pack for UI sounds), it won't be read there, as it's meant for textures.

Another thing to be wary of is sometimes using the wrong audio format in your SCD. **OGG** for BGM/music; and **WAV** for basically everything else, like SFX/sound effects. While it's less likely now, mismatched type and format can still potentially cause silent audio in the game, so try this first.

Additionally, if that step failed, try a different method for conversion to OGG before importing it into VFXEditor. Some programs and websites will modify the file in ways that VFXEditor doesn't expect - such as inserting additional metadata or working with a nonstandard encoding - and may cause sounds to be mute.

To that extent, you could choose to clear all metadata from your sound before attempting a conversion, just to be on the safe side. Audacity can do this with *Edit > Metadata Editor*: press the *Clear* button to remove all existing entries, and then *OK* to save your changes. Select music players will also have similar options, but may not catch some entries with custom names (like if you got the sound from an unsanctioned website and they were very adamant about shoving their name everywhere), in which case I like to use a program like MP3Tag to edit these.

If you've tried all that already, and also tried relogging or restarting your game, look through the rest of the troubleshooting points.

If you still have issues, I am always happy to assist you personally.

TexTools is having issues recognising an .scd path.

At current (granted, I haven't checked in ages), there seems to be a general error when using mixed-case .scd names within TexTools, as it only accepts lowercase. (Example: you want a file path to be saved as *music/ffxiv/**bgm_season_easter**.scd* and not *music/ffxiv/**BGM_Season_Easter**.scd*.) This issue doesn't exist in Penumbra or XIVLauncher plugins, and direct pulls with resource-reading plugins show their original mixed cases, although Penumbra does automatically change it to lowercase on refresh. Please keep that in mind if you choose to work in TexTools.

TexTools still doesn't recognise it or won't let me swap paths!

I've heard that if you're working with BGM outside of the index files located in the base *ffxiv* folder (like those specific to other expansions), it's unsupported by TexTools and you'll have to use Penumbra. I don't personally futz with anything outside of Penumbra nowadays, so if you have further TexTools issues, please visit their Discord server and ask in one of the designated help channels.

I crashed after I turned my music mod off, disabled an option, or removed it from VFXEditor while it was still playing.

FFXIV will remember the last played music SCD for somewhere between 15-30 seconds after you play a different one. This is so it can go back to its stopped position if you return to it quick enough. Playing another music SCD (that isn't the original or the second) will replace the original stored SCD with the second. Therefore, in case you need to turn off a mod that modifies music or you need to do a quick edit, it's recommended to swap unique BGM twice before playing the original affected one again.

Or you can use the Orchestrion plugin's **1 None** entry; this completely stops the BGM and lets it restart. Be sure to let the music fade completely and wait about five seconds before doing anything else. Otherwise, once the BGM reaches its (original) cached loop point, it will fail to read/check against the SCD properly and subsequently crash your game. This doesn't apply to something like non-looped SFX. Those can be safely turned on/off, swapped, or edited once the sound has completed. For looping SFX, it may be safest to swap zones first (to one that doesn't use it) before doing anything else to it.

When I replace PvP, Eureka/Bozja, or 50/60 alliance music, strange things happen!

If you plan on replacing a BGM with the **DynamixStream** property (where it contains both a standard and combat BGM in the same track), keep in mind that a straight swap using "Replace sound file" in VFXEditor won't have the intended effect. As of March 10, 2024, importing any audio with six channels turns the entire file into a garbled mess of static after the conversion process. An imported mono or stereo file will play, but also will not dynamically change in combat. However, four-channel audio is now functional and works almost the same as six-channel audio in these cases, so this is the recommended approach at this time.

Also, please note that, insofar as I know, you can only get the intended result if you replace areas/BGM that natively support dynamic combat music. For example, 50/60 alliance raids and Bozja are possible, but standard dungeons and higher-level alliances that don't use it will only play the first two channels of your audio, regardless if all the settings are correct. I'm still trying to ascertain if this is a limitation that can be bypassed only with a dedicated plugin or there's metadata in the SCD that can circumvent it. But considering that even doing a straight replacement of something like the dynamic LotA theme onto the Tower of Babil BGM will have the same result, it's likely the former case.

In the event you're curious, this limitation is defined in *bgm.exd*, one of several datasheets that are the bane of many a modder's existence. One column in this EXD is *SpecialMode* - 0 is standard BGM, 1 is combat- or special-status-dependent BGM, and 2 is for mount BGM. There's also one set to 3 (*music/ffxiv/BGM_EX1_PvP01.scd*), although I can't place what this one does in particular right now.

Key	0: File XivString [0]	5: DisableRestartResetTime Single [4]	1: Priority Byte [8]	6: SpecialMode Byte [9]
20103	music/ex5/BGM_EX5_Event_25.scd	90	1	0
17	music/ffxiv/BGM_Rade_01.scd	90	1	1
146	music/ffxiv/BGM_Con_Bahamut.scd	90	1	1
180	music/ffxiv/BGM_Con_CrystalTower_01.scd	90	1	1
202	music/ffxiv/BGM_Con_Bahamut_v2.scd	90	1	1
218	music/ffxiv/BGM_Con_CrystalTower_02.scd	90	1	1
220	music/ffxiv/BGM_PvP_Battle.scd	90	1	1
247	music/ffxiv/BGM_Con_CrystalTower_03.scd	90	1	1
327	music/ex1/BGM_EX1_Alex03.scd	90	1	1
347	music/ex1/BGM_EX1_Makosen_Nomal.scd	90	1	1
393	music/ex1/BGM_EX1_Makosen_Nomal02.scd	90	1	1
500	music/ffxiv/BGM_PvP_Mogi_01.scd	90	1	1
524	music/ex2/BGM_EX2_EU_Field.scd	90	1	1
580	music/ffxiv/BGM_Season_XmasCho.scd	90	1	1
718	music/ffxiv/BGM_Season_XmasCho_02.scd	90	1	1
722	music/ffxiv/BGM_PvP_Battle_02.scd	90	1	1
772	music/ex3/BGM_EX3_MYC_01.scd	90	1	1
791	music/ex1/BGM_EX1_Hukko_concert.scd	90	1	1
61	music/ffxiv/BGM_Ride_RChocobo.scd	90	1	2

Note that the ones marked with *SpecialMode* of **1** (possibly **3**) here are the only ones in current existence that can be manipulated for dynamic BGM.

I'm trying to substitute an orchestrion song for another BGM and it's not working.

Since I see people often wanting to make orchestrion songs into BGM: in order for it to successfully play, you need to either export it as OGG and replace it in your intended BGM's file, or change your bus number and priority number to reflect that of BGM (i.e. instead of bus of 16 and priority of 194, you should make it bus of 1 and priority of 180). If you're doing the latter, I would also uncheck **BusDucking** in the *Tracks* tab. As the base file is mixed down to mono with a considerable loss in fidelity and reduced overall volume (which makes sense for its medium), I recommend finding a proper version of it instead, if you're able. Unless you like that sort of thing. No judgement.

I replaced a non-looping BGM with one that loops and it cuts out early.

If your looped file cuts out after a while when replacing a non-looped file in an SCD, try to match sample length of your custom file to the one you're changing. Samples values can be checked through Audacity or similar programs. You may have to cut extra audio or add in some silence as a buffer. I have not personally tested this, but, in theory, it should alleviate any issue that could arise.

If it's still causing problems, message me and I'll do my best to assist.

The game stops playing sounds altogether when my BGM reaches the end of the loop.

This usually happens as a result of an incorrect seconds value specified at the loop's supposed end, which ends up defining a point beyond the actual file length, mostly with songs that go from the end of the file right back to the start. VFXEditor may sometimes show a value that isn't properly aligned, especially if you're defining *LoopStart* and *LoopEnd* manually in the metadata of an OGG. In this case, type in an absurdly large number into the end value and it should correct itself. Remember that the number shown is an approximation and not down to the exact second/millisecond, but will still function the same. It won't do auto-adjustments with WAV, so you'll have to manually double check where your file ends.

The volume of my SCD is really low and/or it doesn't seem to change when I adjust it.

There are a lot of factors that could play into this.

As always, if you're changing the volume of standalone BGM to test (as in not attached to something like an emote or VFX), make sure the sound itself has completely reset first, else it's likely you're still hearing the cached volume setting. Make sure that the sound has actually been modified as well, such as having the proper path set in *SCD Being Replaced* and the *UPDATE* button pressed, or the mod itself containing the updated file with no existing mod conflicts.

If those are fine and dandy, double check the *Volume* value you put into the SCD in the *Sounds* tab. This generally has a range from 0.0 to roughly 2.0 and cannot go beyond that. The other *Volume* fields in the SCD are limited to 1.0 and base themselves off of the *Sounds* tab one. Manually editing the sound itself to be louder/softer in an external program is the way around that, or you can choose to use ADSR in the SCD's *Tracks* tab if you don't care about some the sound being a little blown out due to clipping or having some extra effects tacked onto it. Sometimes, the *Volume* entry in the *Tracks* tab will be set lower than usual as well, so be on the lookout for that.

If this still doesn't fix your issue, look at the bus number. Whatever it uses, double check what you set its associated volume in FFXIV's settings menu to or if you accidentally muted that channel. For example: **1** and **16** will be affected by **BGM**; **2** will be affected by **Sound Effects**; and **17** will be affected by **Performance**.

Still not doing it? Check how your sound is being applied. If it's attached to a TMB/PAP via a *Sound (C063)* entry, see if *Use Min Max Range* is checked or not. You can try unchecking this if you don't care about proper sound falloff/distance fade (i.e. a sudden cutoff at the maximum range), or check it and adjust the *Min Range* and *Max Range* values in the SCD. You may also look into changing *Range Volume* and *Near Fade Start/End* (see Appendix C for details on all of these). If it's attached to a VFX, this will use *Min Range* and *Max Range* by default, and base itself off of the relevant emitter's binder.

I can't get my music to continue playing in some instanced ARR content. It always gets muted after combat BGM interrupts it!

This is a technical limitation. For some reason, the FF devs decided to keep early ARR music non-looping. It's hard-coded. (Remember when there was a "bug" that made them loop for a brief few days before they patched it? *I sure do*.) This cannot be reversed without doing hex edits of .exd files, which is incredibly difficult, would require constant updating, may venture into foot-gun territory, or is downright infeasible. I share your pain. To that end, the best you can do right now is use the Orchestrion plugin and play something over the original music, or replace the original with your own music and play that through the plugin. Copium at its best.

Is it possible to make mount BGM unique and not shared with another?

As I've seen a couple people ask about replacing mount BGM in the case where it's shared with other mounts, the answer is sort of, if you attach it to the mount's VFX and set the SCD's bus ducking data to mute the BGM channel. However, do note that VFX will not keep the last-played position of sounds, so every time you dismount it'll start the BGM all over again. Bearing that in mind:

If the mount has an existing VFX, you can slap the BGM on that by inserting the path in one of the emitters; usually, *Emitter 0* is a safe choice. See method two in the examples section on how to go about this if you're unfamiliar with VFX.

If there's no VFX, you'll have to do an IMC edit to force one. Give the mount a VFX ID (an ID of **1** is usually fine) through Textools/Penumbra. Then use any sort of looping VFX as the *Loaded VFX* (easiest to borrow from another mount that has one) and set the *VFX Being Replaced* path as *chara/monster/m[monster ID]/obj/body/b[body ID]/vfx/eff/vm[vfx ID].avfx*). Example: with a VFX ID of **1** in the IMC metadata, a monster ID of **6008**, and a body ID of **1**, the path becomes *chara/monster/m6008/obj/body/b0001/vfx/eff/vm0001.avfx*. Assuming you don't want the random VFX actually showing up on your mount, and only the sound, you can probably just change the *Revised Scale* in the main *Parameters* tab of the VFX to something miniscule and non-zero (e.g. **0.0000001** or whatever). It'll effectively not exist, even though it does. Again, see method two in the examples section on how to go about this if you're unfamiliar with VFX.

Well... what about sticking the music on the mount's PAP file, then? Bad news, everyone! A mount PAP has multiple entries for each possible state of the mount - flying, jumping, idling, etc. Even the player-based mount PAP will have at least two. Trying to place a BGM on it will then have to be restricted to that specific part of the animation. Ergo, every time the animation changes, your BGM just starts all over again. TMBs are even worse for this, as the BGM won't actually cancel with the *Stop On Animation End* checkbox like its PAP counterpart.

For those curious about why you can't just directly make the BGM custom and don't mind a technical explanation: the pertinent information for what governs mount BGM is a specific flag in the ***mount_0_en.exd*** file (and the other three language-specific files by replacing *_en*), which uses an index number to tell the game what to use. In this sense, you're limited to what exists in-game, unless you also edit ***bgm_0.exd***, add in a new entry, and then point your custom BGM path to that established path.

Two things with this now:

> **1.** Currently, this path addition is difficult/tedious to achieve, as the game reads this file strictly at startup (meaning TexTools injection only) and you'd need to dive into hex code editing. There are very specific values that have to be edited to avoid outright crashing, and then you need to manually change any offsets so that they match.
>
> **2.** If you chose not to do that and work with the indices/paths you have, you would have to manually update the EXD *every time* the developers made a change to it. Ergo, if you chose to stop playing the game at some point, anyone who uses the mod would be stuck with it being nonfunctional after a server-side change and would crash their game if they tried to use the old EXD. These EXD datasheets also run this risk of sending *invalid data* if things don't match - which isn't really an issue for the BGM ones, honestly, but I'm throwing this in here as a warning. All that said, a plugin would be the best bet for this (of which there are none for this datasheet yet), but it would still need to be regularly updated for every patch.

It's almost like they don't like fun or something. /s

Some SFX get interrupted whenever they're played repeatedly. Is there a way to change this behaviour so I can always hear the full sound?

You might try the solution listed in Appendix E at the section labelled *First Condition*. I'd advise doing it sparingly, as it could become difficult to discern other sounds, depending on what you're trying to edit. (Or could potentially even cause a crash if placed on a highly spammable skill or emote, such as certain lalafell voices that tend to pollute the air in Limsa. While it'd probably take an exorbitant number of sounds to do that, it's better to err on the safe side.) This is especially true for sounds in combat. I would also recommend taking the custom path approach as well if you're borrowing sounds from sources like enemies, in order to preserve their integrity.

My music doesn't want to loop nicely, no matter what I do!

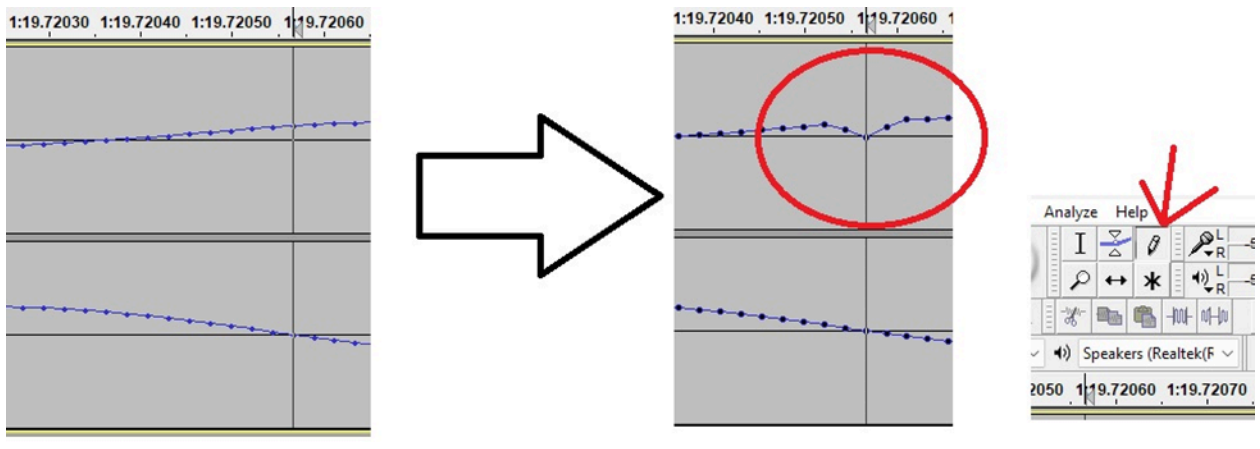
Same. For real, though, I have a series of related solutions that may help with this. The quickest way is to try signed 16-bit ADPCM. It makes the audio lossless and loop points should be exact; however, this will triple/quadruple your file size, so this is not recommended for long audio. You may also find that the editor handles the file type strangely in some cases.

Barring that, here are a few options to mess with loop times, if you continue with OGG:

> - Check the section on preparing audio with FFmpeg. This is your first go-to if you don't mind foregoing a comfy UI for a text interface.
>
> - Nudge both your *Loop Start* and *Loop End* values a bit. Usually, if your OGG was converted in a standard manner, your *Loop Start* is more sensitive to value changes, especially if there's less going on there [in the waveform].
>
> - Adjust the length of the audio by a small amount, whether by trimming some of it or by adding extra silence/sound at the end. (If you have one of those songs that fades a second after the loop starts again, maybe consider duplicating a part of the earlier loop start to splice onto the end and give yourself more wiggle room.) If you've added audio to the end or start of your file and it'll fall outside of the loop, try adding post-effects to it like pitch shifting, volume boosts, or fades, something that will change the audio's size on the disk when exported. I believe this last bit is the least likely to help, but still worth a shot.
>
> - Adjust the volume slightly outside of VFXEditor. Even a +/- 0.1dB change can help.
>
> - Change audio quality and/or frequency.
>
> - Try a different spot for looping, if at all possible. If it ain't working, don't force it.

Since VFXEditor/the game uses variable-size OGG pages to establish where to split its seek positions, these methods can allow the program to slightly change where these pages are split. Less technically, if you edit the overall size of a file, there's a chance that loops will turn out cleaner because of how the program will read how it should separate the sections of the file. The FFmpeg method can work around this to a degree; although you will still have some amount of finagling to do, it will be much less than normal.

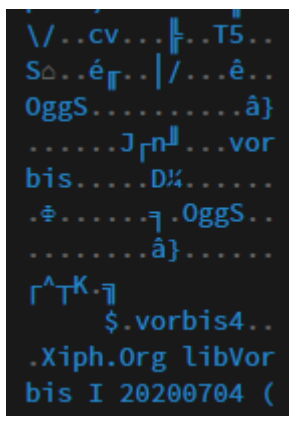
Finally, if you went the route I suggested at the beginning of the document, with finding zero crossings in Audacity and placing loop points there, I have a rather sneaky way to give yourself more of these zero crossings instead of relying on where the music/sound has it by default:



The left-hand side is the untouched file. I've zoomed in on the track just enough to where Audacity places these little dots on the waveform. Using what's called the *Draw Tool* - shown at the right, the pencil icon - you can actually *move* these dots in your favour and make your own zero crossing instead of hunting for one in a spot you may not like. Generally, what I do in these cases is shift the dot I want to the zero line, and then I also shift a couple dots on either side of that dot I moved at the start in a curved pattern so it's less likely to be apparent when you play the music. (Think of it as a smooth transition and not a sudden spike in a music note.) I don't recommend this as your first or even second option, as there's a slim chance you'd notice something off when you listen to it, although it's not as big of a problem as you'd think, provided the dots you move are close to the zero line to begin with. It's just something to consider if the loops are driving you nuts as they are. Mostly, you'll resort to this if the instruments are more live than programmed or if the music wasn't intended to be looped.

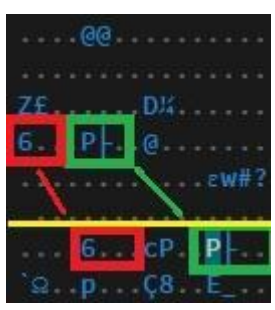
Addendum: *if* you wish to go insane, there's a possible hex solution, but to put it simply it's not entirely viable. Still, I'll leave this here if you want to be educated.

There's a small section right before the first *OggS* in a standard custom Vorbis OGG-based SCD:



Aside from the very first two values, these appear to be sample numbers (i.e. how the OGG determines where to establish points to seek to), determined/seen in the 24- or 32-bit integer fields of your hex reader. Changing these does manipulate loop points to a degree, but it's still not accurate in the sense that you could change a specific one to exactly the sample you want and it'll make the loop perfect. The seek table has a very specific reason it populates with these particular values, although changing them won't break the sound functionally. Additionally, VFXEditor's interface displays loop points in *seconds* now, and these seconds don't correlate with what a program like Audacity will show at the same time (because VFXEditor works on an estimate of some kind).

So, where does that leave us? Well, staying in hex mode, we can manipulate our loop points ourselves without messing with the actual interface:



Right before the start of the sample metadata (where it starts below the yellow line) are a few determining values, but the important ones for us are three lines above the sample data. The red block is our loop start sample value, while the green block is our loop end sample value. Note that even if you try to make the loop start/end values your exact samples, this will round up or down to the nearest sample value established in the sample metadata block.

Now you see why VFXEditor's more recent change makes more sense, hopefully. It's not VFXEditor forcing you to specific values, per se, but how FF14's internal code in tandem with OGG handles the samples being set; and so VFXEditor has chosen to reflect that forced rounding in its interface.

If you're confused on where all this is found in the mess of hex, look for a series of four-column bytes above the aforementioned *OggS*. (These can be pairs of bytes or even possibly sets of three in each column, depending on overall length of the sound, so be aware.) Where the sample metadata ends is right before this *OggS* string, while where it starts is always the very first **00 00 00 00** when you're reading from the bottom up of that metadata block. With FFXIV native files, this OggS header data doesn't exist, so try to orient yourself from either the top of the file or where you see the columns end. If the 24-bit integer field loses its gradual incrementality, you've gone too far.

I'm seeing HCA as an audio type now in the editor. What is it?

In patch 7.5, Squeenix decided to incorporate a new format into their SCDs - **HCA**. It's a proprietary sound format that's used in games made for the PS2 and most things by Criware. A plethora of short sounds (like jingles, voices, and things in *SE_common*) have been converted into HCA instead of their usual WAV format. This is likely to save space, since the compression in the new format is stronger, albeit at a small loss of quality that will mostly

go unnoticed. Since it's "new" - at least in the scope of FFXIV - it's possible that some plugins or programs will not play the audio back properly (or potentially crash, but they technically shouldn't). Keep that in mind if you choose to work with HCA.

Do SCDs extracted from other SE games work?

They do! (Or at least the ones I tried.) The metadata won't be exactly the same, and it may show as Unsupported because of that, but it's very much plug-and-play. Even bus numbers are almost equivalent. For the FF13 trilogy, you can use WhiteBinTools to do the extraction. For FF13-2, BGM in particular has entries in *filelist_scrc.win32.bin*, which draws from *white_scrc.win32.bin*. An example argument in Windows' command line tool would be: *.\\WhiteBinTools.exe -ff132 -uaf "filelist_scrc.win32.bin" "white_scrc.win32.bin" "sound/pack/8000/music_handsnow.win32.scd"*. (This assumes you have the extraction tool in the FF13-2 game directory containing all the *.bin* files. **.** gives cmd proper permissions if you need it.) Grabbing the file list for perusal would be, for example: *.\\WhiteBinTools.exe -ff132 -ufl filelist_scrc.win32.bin*. Use *.\\WhiteBinTools.exe -?* for a full list of commands and other examples.

Additionally, in FF13 trilogy files, a ***.wpd*** filetype exists: it's a collection of sounds with additional header data that's used to reference each sound. You can safely remove that header data with a hex editor and the SCD will open in VFXEditor as normal - more or less, remove everything from **WPD** up until **SEDBSSCF**. However, the extra WPD data references different sections of the SCD, so anything past the first sound will be unreachable. The good news is that each section starts with the same identifying "magic" text (**SEDBSSCF**); in other words, you can crop specific parts to reach the rest of the sounds.

Side notes:

Dirge of Cerberus uses *.at2* for SFX and *.at3* for BGM. I believe these are ATRAC encodings, although decoding it seems to fail.

FFX/FFX-2 remaster uses *.fsb*. This is decodable with VGMStream/Foobar2000.

FF Type-0 HD uses *.scd*. Unfortunately, they're stuffed in a *.pac* file, and the decoding required is not accessible (requires signup to a defunct site). If you happen to have the *.bms* script, I would really appreciate a copy! Or if you have an alternate way to extract the files, do let me know. Mostly, I'm just curious about how the SCD is used, rather than the sounds themselves.

My issue isn't listed here!!!!!!!!!!!! HELP???????

Mostly, I'm not aware of issues unless I have them myself or see someone talking about it. However, you are more than welcome to message me for direct support - provided it's not too often, as I do still have a job and many other things to do. There are also plenty of Discord servers available for specific help, if you're not one for private chats or want to figure things out yourself. I won't list all of them here, simply because there are too many, but the main ones will have links somewhere within their interface or - more in the case of solo creators - their own Twitter accounts. I encourage joining Students of Baldesion (this link should be permanent now, but if it's not, just DM me) not just for help with sound modding, but for advice on all types of mods. They're a very helpful bunch. As I'm also a member there, you can ping me on the server if you need to (*@caffeinatedgator*).

Changelog (because things get changed a lot)

26/08/26: Some formatting changes in the appendices to make it a little more readable, perhaps.

11/08/26: Updated appropriate sections of the document to match a GUI update. (To do: see how the recent code changed loop handling.)

31/05/26: Fixed some sections since WAV no longer works for BGM.

06/05/26: Adjusted some *Bus Ducking* mentions for congruency with the new fields. Added stuff about the new HCA format, where applicable.

18/01/26: Adjusted the aforementioned sections and made some descriptions clearer, as well as supplied some basic screenshots.

08/01/26: It's a new year and there's new stuff. I've preemptively added some data on the *Bus Ducking* and *Extra Desc* tabs that should show with the next VFXEditor update. These should give you more freedom to adjust certain behaviours, although they're still mostly unknown due to their newness. I'll try to make it less confusing as soon as I'm able.

07/09/25: Added a section on using the FFmpeg standalone codec in order to further manipulate OGG page structure. Thanks to *floppy drive* on Discord for mentioning this method. Let me know if this helps in any way. Also fixed/removed some old comments about TMB leaking and the funny moon plugin, since those no longer apply; and updated a couple screenshots for C063 to be more accurate.

05/01/25: Added a troubleshooting point about the "Unsupported" error brought about by custom WAV and SCDs created in non-native programs.

02/01/25: Reworked the document a bit to remove that clunky primer example and make each of the other examples self-sufficient. Tried my very best to improve some readability and remove crossed wires. That did mean a lot of repetition in examples, but it's probably fine.

04/10/24: Clarified in a couple spots about a common misconception on custom path folder creation. Removed a small section in the looped emote section about it so it wouldn't cause additional confusion. In a nutshell, manual folder creation only matters if you have a *finalised, exported mod in Penumbra* and you want to edit it further. Any path established in the *Being Replaced* field in VFXEditor is temporary and isolated to the editor, meaning it cannot read local paths unless you tell it so by clicking the magnifying glass and searching for it in the Penumbra tab. (Note that these changes are also temporary, so clicking UPDATE doesn't actually edit the mod's file or create it in the spot you specified; it just provides a temp file for testing purposes. These changes will be gone once you clear the tab/editor, although the temp file will remain in the folder the editor uses, specified in its settings menu.) I should put this explanation elsewhere, once I find a good spot for it. :^)

01/09/24: Some hex nonsense for another slightly improbable solution in the "music doesn't want to loop nicely" troubleshooting point. Or at least you'll see why VFXEditor behaves with loop times as it does, and how it's not possible to change that. Please don't bombard the Github with complaints. Small pops of colour for subheadings. Reordered stuff in the tracks appendix so it's alphabetical instead of frequency-based. Minor addition to the attribute appendix. Added an appendix on Lua conditions. Deleted the TMB/PAP/VFX appendix and made it an example section instead because I'm getting *a lot* of messages about how to add just SFX and not music, even though it's the same concept. Y'all want to be hand-fed. I get it. :salt:

18/05/24: Fixed an erroneous mention of *Loop* in C063 being restricted to three states, when in fact it can be set as *Duration*. I have seen the error of my ways.

12/05/24: Added a small-ish section on encoding size comparisons, alternate methods of conversion, and alternate options for SCD implementation or lack thereof. Tiny rework of the loop point troubleshooting section for clarity's sake.

03/04/24-04/04/24: Some minor testing updates for content in *Layout*, specifically *Use Dir First Pos* and some *FAC* settings. Added a section on the newer stuff in the *Audio* tab. Changed info on *Local Number* to reflect user research.

10/03/24-14/03/24: Extensive rework of the document. A lot of information has been reworded, removed, or changed entirely to reflect VFXEditor's GUI updates in v1.8.2.x, or just because I've decided to use my brain for once. Four-channel combat-based BGM is now viable. Did a write-up for randomising BGM for *non-BGM replacements* (I'm sorry - I really wanted that to work, but I'm stumped, so you might look into third-party plugins or just use an external player). I've also added a lot of colours, pictures, and indentations to bring attention to certain points or make things potentially less obtuse. Gone is the yesteryear of writing being fully black and pictureless. Soon, novels will be just pictures and tiny bites of words. ...No, wait, those are films with subtitles. There's also a font change because I can. Some things haven't been given a facelift, but I'm getting there. This all took a **lot** of time. *Y'all better be grateful*. (I'm joking. Kind of. If I actually used my Kofi, I'd ask for tips or peppermint mochas, but I don't because taxes suck.)

01/03/24: Explained about limits in *GroupRandom*. Will probably do a write-up soon about random BGM per play. Note that you must be on the VFXEditor beta build to try it. The main branch isn't yet at that stage.

04/01/24: Did minor research on the *Unknown* tab for Fixed Volume/Bypass PLIIz. Possibly confirmed what First Inactive does, at least in relation to Height.

14/12/23: Added a fourth example to cover sound cropping. Moved the example section to be congruent with its placement in the table of contents.

12/12/23: Adjusted some explanations to be less obtuse and confusing (hopefully). Eschewing obfuscation and espousing elucidation.

29/11/23: Added a couple mentions about the vgmstream player on the interwebs. Intial tests seemed good, but I got a little ahead of myself. :)

12/11/23: Did some more research and changed/added where applicable (mostly Layouts/Tracks).

07/11/23: Added another (hopefully) simple example on sound replacement for existing mods because I hear a lot of secondhand comments about this document being a little confusing. I'm an English major and novelist. I like to word vomit. 本当にごめん.

03/11/23: Added a section on *Upper Limit* use in the *GroupRandom* category.

10/10/23: Added a series of examples to try to simplify workflow for the end user. This is in-progress and subject to change.

29/09/23: Rectified notes about some .scds done with outside conversions being unreadable. This has now been fixed in the editor, albeit marked as unsupported. Be wary when messing with these files further.

12/09/23: Added a troubleshooting section about why people are getting weird playback issues in-game with conversions.

26/08/23: Cleaned up some of Appendix F to be more intuitive and inclusive.

21/08/23: Included notes about how *Volume* fields are handled in VFXEditor, mostly that there is a set minimum and maximum value that the game allows on its own. There's also an additional troubleshooting point now for anyone who has a hell of a time getting their .scd loops right.

24/07/23: Added headings in the document structure to allow for jumping in the outline menu, upon request (which I should have done in the first place, but (complete honesty) I had no idea it existed).

11/07/23: A small addition to the FAQ section to reflect some brief testing found in Appendix E.

07/05/23: Continuation of research in various tabs.

27/04/23: Small edit for *Sound (C063)* to reflect a recent VFXEditor change.

13/04/23: Expanded on the Min Range and Max Range values in the Layouts tab. This should help for anyone who needs to clearly define the distance at which sounds play.

10/04/23: Made the troubleshooting section more legible by adding subheaders, and slapped on a couple small points here and there.

07/04/23: Included a small segment on multi-channel audio for the built-in player. Changed other minor details regarding 6ch files to reflect testing, but overall it's the same, for the time being. Added a troubleshooting point about making BGM unique in places where it uses one that's shared. tl;dr - suffer.

04/04/23: Changed small parts of the document to reflect the April 4th VFXEditor update.

02/04/23: Removed the rickroll meme from the changelog link. It's old but I still love doing it. I've also tacked Vulonsulin's Google Drive spreadsheet collection onto the .scd resources section of the guide. "Credit's not necessary." Gurl, please.

01/04/23: Decided to do a write-up for .ttmp2 creation because I saw a lot of modders on XMA saying they had no clue on how to do it. That was also me before yesterday.

30/03/23: Finally used my pea-sized intellect to discover there's more than one expansion folder you can search. Updated the FFXIV Explorer listings to reflect my galaxy brain moment.

29/03/23: Compiled some late-night-into-"oh, god, is it daytime already???" .exd research. Probably reached a dead end there with the errors I'm getting and searches turning up very little.

27/03/23: Did a quick write-up of .scd for .pap as well as applying a custom path there.

24/03/23: Did research on *Voiceline (C053)* and put the results in the corresponding section. Fixed incorrect information in a couple areas.

22/03/23: Edited the document to reflect the newest Scd Editor update. Additionally, shifted other information around, deleted redundancies, and corrected potentially misleading points. Gave "all systems go" on looping field BGM. Probably was fine in the first place and was just a specific subset in dungeons, so *that's on me*. ~~I'm sorry~~. If something seems contradictory, please contact me because very often I'm blind and don't realise until a month later. お疲れ様ッス.

18/03/23: Included another appendix on changing .scds (and related options) for .tmb and .avfx files. Fixed bookmark links. How long has that been broken? *I'm sorry*.

16/03/23: Shifted large chunks of tab info into their own sections to avoid clutter in the scope of the guide's original purpose. Added more in the appendices.

15/03/23: Added a troubleshooting point about converting orchestrion songs to BGM. Fixed my well-intentioned, but ultimately incorrect, point on the BusDucking attribute in the Tracks tab. I'm welcome. Included an appendix on mount sounds and SFX modulations. Started two appendices on the *Tracks* and *Layouts* tabs. Largely WIP, mostly because I continually crash to the desktop and half the time I want to chuck my computer out the window.

09/03/23: Added more description for the Acceleration metadata and fixed some erroneous information in BusDucking. Included a couple more troubleshooting points.

27/02/23: Added a small paragraph in the intro that mentions the oft-linked guide on editing a dance mod's music. Link included.

26/02/23: Small clarifications here and there. Added a segment on the DynamixPlus attribute (tl;dr - layered .scd for smooth combat music transition).

23/02/23: Shifted sections and moved research to a separate document to reduce clutter. Added a minor bit about using Anamnesis to find certain sounds.

20/02/23: Did a tiny bit more research. Added a minor bit about the ResLogger2 plugin and its use for finding .scds. I'm considering doing a fair bit of condensing later on for the guide as a whole, or at least appending it with a simpler walkthrough.

02/02/23: Blocked in a bunch of XIV Explorer path descriptions to eliminate some guesswork. There is still a lot of guesswork, though. :)

30/01/23: Added some preliminary (very technical) VFX research. Started working on the full list of .scds in game. I am very insane. [Both of these segments have been moved - links in the appropriate sections.]

27/01/23: Minor clarification on the singular troubleshooting point. General apology if things are extra-super scuffed (and if things change, the document will be updated ASAP).

22/01/23: Added another method for finding loop points with bytes. Established that Interval values seemingly have no tangible effect. Fixed grammar mistakes because I'm a bad English major.

Concluding remarks (a.k.a. You've reached the end! Finally.)

Thank goodness. I'm exhausted.

Hopefully, this guide was of some help to you and wasn't too much of a slog. It turned out a lot more technical than I thought it would, but I felt like I needed to cover bases and clear up any potential questions that could arise. As stated in the beginning, if you have questions, concerns, corrections, or something's on your mind, head to @gatornaut on Bluesky or #taco1932 on Discord. I do have other places I'm active, but there's less guarantee of a quick response. I may be invisible on occasion, but I promise I'll reply when I'm not working or planking in bed.

Thanks for reading!

Anyway, have one last meme.

(Also, random tidbit: the chanting in the Ramuh track is also sampled in a Firefly episode. I don't know why you'd want to know that, but now you do.)