Folder to SFZ Converter

Instruction Manual

April 2016 – Version 0.4

Versilian Studios LLC.
In Collaboration with Blainicus

http://vis.versilstudios.net/

Notice: Requires Java 8+ to run. You can update your Java here.

General Overview

This utility was originally internally designed to rapidly and effectively map large numbers of samples to the SFZ 2.0 format. We decided to make it public to help make the creation of sfz instruments easier and faster for everyone.

Operation:

- Unzip the contents and run FolderToSFZ.jar *(platform agnostic)*.
- Fill in the fields *(see UI Functionality below for details)* and select or create correct dictionaries.
- Press Execute.
- If there are issues, view the log found under logs *(by session)*.

See a video walkthrough of using this program here.

UI Functionality:

Ampeg Attack- Specify the ADSR value for attack, in seconds. Recommended between 0.01 to 0.05. Values over 0.1 will have a noticeable effect on the samples.

Ampeg Release- Specify the ADSR value for release, in seconds. Recommended between 0.1 to 0.3 for sustaining and greater than 3 for non-sustaining.

Volume Norm- Primitive balancing- automatically adds volume to quieter velocity layers, spread linearly. If you specify 12, for example, it will increase the bottom vl by 12, the middle by 6, and the top by 0 (theoretically). Setting to 0 will do no balancing.

Round Robin/Rand Robin- If using RR, you can use either round robin (a locked sequence cycled through, more useful for more RR or specific patterns) or random robin (a set of options defined by probability, more useful for fewer RR).

Notation Dictionary File- Four default notation options are provided, C4=60, C3=60, 60=60, or a wholetone scale starting on 0 example. Additional dictionaries can easily be created or customized in text editor of choice.

Velocity Dictionary File- There is one main dictionary file for velocities that includes a variety of methods of describing velocity. See below for accepted formats.

Velocity Algorithm File- By default, your velocity layers will be divided linearly (evenly) across the 0-127 range. However, you can create or specify a custom distribution using an algorithm file.

Input Folder- Select the folder of samples to map. Does not browse sub-directories!

Output File- Navigate to the desired folder (the folder should be "above" the samples directory) for your instrument file and create a new file. If not filled, a default file will be created in the sample directory. You can configure path information if you decide to move the file using the opcode "default_path=" under a <control> header.

Execute will run the converter. Any errors will be displayed and all information regarding the export will be logged in detail in the logs folder.

Filename Arguments:

Type	*Accepted Arguments*	*Examples*

Key	MIDI Key Name
	C4=60
	C3=60
		64
		C#3
		C#2

Velocity	Traditional Symbols vel=*x* v*x* vl*x* dyn*x* adjectives	pppp-ffff vel=2 v2 vl2 dyn2 quietest/softest-loudest
RR	rr*x*	rr1

Customization

The real heart of the program is the customization. You are able to customize the various dictionaries and mapping methods to fit your needs and own arguments.

Custom Dictionaries:

Key and *Velocity* accepted arguments can be modified via the text file **dictionaries** in the corresponding folders.

To create a custom key dictionary, create a new text file in NotationDictionaries with the format Notation=KeyNumber (the *keynumber* ranging from 0-127, corresponding to the MIDI standard). For example, lets say you have an instrument mapped with integers (1,2,3, etc.) for key names, sampled wholetone starting on MIDI note 60 (C4). You would start your dictionary with 1=60, then 2=62, then 3=64, etc. See "Piano_Wholetone.txt" as an example of a library mapped up the *wholetone scale* starting on A1 (21).

To create a custom velocity dictionary, I suggest editing the existing dictionary file under VelocityDictionaries. Make your entry in the form of Label:VelocityOrder. Higher values are higher velocity layers, but not exact. For example, you could use the tags v2, vl3, and f, which has the velocity ordering 2, 3, 6, and it would be treated correctly as three velocity layers in the correct order. Add your new system where it fits into the order (i.e. a new entry will probably start best as x=1).

No other arguments are accepted for RR. It is typically very easy to use a mass file-name editor to comply with this standard.

Precautions:

- The maximum number of velocities allowed is 12 using v*x*, vl*x*, vel=*x*, or dyn*x* notation. Traditional notation (pppp-ffff) allows up to 9 velocity layers.
- If no velocity algorithm dictionary is selected, it will default to an internal calculated linear distribution.
- Do not feed a folder with multiple mic positions in it at once into the program, it will crash (instead create custom folders and create a new instrument per each mic position for the time being).
- Do not create a velocity dictionary entry that uses only an integer or have extra lone integers that do not correspond to key ranges if you are using a key#=key# (e.g. 60=60) system of key mapping, you will confuse the system.
- Make sure you add "rr" before the RR argument of your filenames.

After Execution

After exporting the project, there are a few steps you can take prior to distribution or use. Below are some resources and general tips to help improve your project.

Verification:

First and foremost, verify that the execution as successful. Ensure that the file was created and that it is not empty/near-empty. At least one <region> should have been formed. Occasionally there will be errors or invalid arguments that cause issues, in which case, double-check to make sure you don't have incorrect arguments or an extra integer line that is messing up a read line. Remember: you're smarter than the computer, it can only infer what it is reading, it doesn't know.

If you cannot troubleshoot the issue on your own, feel free to contact us at contact@versilstudios.net for advisement, including a full report and screenshots, if possible. We are glad to help folks out with any issues.

Manual Processing:

You may decide to manually edit the files without the help of any 3rd party editing tools. If you are inexperienced in the sfz format, I highly recommend checking out the following resources:

- Peter Jones' Webpage (includes good examples and information)
- LinuxSampler SFZ Reference (helpful catalog of almost all sfz opcodes)
- Simon Cann's Book (the "official" guide to sfz)
- Simon Cann's SFZ Tools (for UltraEdit, for manual text editing)

3rd Party Editors:

There are a number of 3rd party conversion and editing tools that support sfz, such as Extreme Sample Converter, so that you can edit your files or convert them to other formats.

Troubleshooting

If any issues are encountered in using this program, feel free to contact us at contact@versilstudios.net. In order for us to help you faster, please include the following information:

- Your *Operating System* (including version)
- Build of *SFZ Converter*
- Full description of issue
- Screenshots or video (consider using OBS//Open Broadcast Software to make a recording of your screen)

This information will help us resolve your issue.

Credits

Concept & Design

Sam Gossner

Programming & Logic

Blainicus

Evaluation

Simon Dalzell (Ivy Audio)

Mattias Westlund

To everyone involved in creating and using this program, my eternal thanks. -SG