Tim Exile

Scapeshift – *UNOFFICIAL!* – User Guide

Scapeshift: Generative Audio Workstation User Guide

Welcome to Scapeshift, a generative audio workstation that allows you to create intricate musical patterns and soundscapes through its unique parametric sequencing engine.

About this User Guide

This user guide was created through Tim Exile's Patreon Community as an unofficial companion manual to Scapeshift. It is currently under development and is likely to contain errors.

Please report any issues or errors in the appropriate channel on the Discord server!

Edited by

Mikael Lundgren aka The Introvert

Authors

Tim Exile, Phommed, Pink Bob, The Introvert

1. Introduction and Core Concepts

Scapeshift is designed as a **collaborative musical tool** that allows you to steer and dialogue with generative systems in real-time. It aims to provide the unexpected creative outcomes often associated with AI in music, but without relying on AI algorithms or the collection/ingestion of any user data. Instead, it offers deep control over custom-designed generative engines, allowing you to shape the musical output according to your taste and preferences. Scapeshift bridges the gap between high-level creative exploration and detailed parametric control.

Scapeshift operates on the principle of **parametric sequencing**. Instead of directly manipulating note or audio data, you control various parameters that define when and how musical events occur. All settings within Scapeshift are **continuously variable parameters**, which simply means they can be changed to a new value in real time–like a volume knob. This allows for continuous morphing between different musical structures–vastly different if you wish.

Scapeshift is described as a **generative groove box** that doesn't use AI. All of the music it generates is based on generative systems designed from the ground up, giving you direct control over parameters or the ability to create from a high level using controlled randomization via the dice icons.

Scapeshift requires a **full license of Native Instruments Reaktor 6** to run without limitations. It will not function fully with the free Reaktor Player.

Scapeshift is available for purchase through **Tim Exile's Patreon store**. By becoming an **Adventurer** tier subscriber on Patreon, you receive a **50% discount** on Scapeshift and Scapeshift-related products, as well as early access to updates and exclusive content. Go to https://www.patreon.com/c/timexile/shop for a list of available products and complete details.

Scapeshift is under active development, with future updates planned. One highly anticipated feature is the ability to **map MIDI controllers** to Scapeshift's parameters for more hands-on live performance capabilities. There is also a strong likelihood of a **Scapeshift version 2** in the future, with upgrade paths for existing users.

This manual will guide you through Scapeshift's features and functionalities based on the information available in the provided sources. But to get the most out of it, experiment with its various features and delve into the detailed controls of each engine to discover Scapeshift’s full creative potential!

2. Getting Started

When you first open Scapeshift, you'll encounter a user interface with several key areas.

Themes

Themes provide a quick starting point for generating music. Scapeshift includes 12 different themes, each focused on a particular musical style. To get started, simply click on one of the theme cards. Scapeshift will then generate a musical pattern based on the chosen theme. Examples of themes include Dreamhouse (laidback house), Wild Drums (polyrhythms), Infinity (ambient textures), and Acid Melt.

Dice Icons and Randomization History

Throughout the Scapeshift interface, you'll find **dice icons**. Clicking these icons triggers a randomization of the parameters associated with that section. Scapeshift keeps a **history** of all randomizations, accessible via the arrow icons to the left and right of the main dice icon at the top. You can step back and forth through your randomization history to revisit previous states. Clicking and dragging a dice icon allows for **partial randomization**, which is very useful for blending elements of different themes or styles. Drag a dice icon a little or a lot to control the level of partial randomization that will be performed.

Macro Knobs

Above the pattern display, you'll find **macro knobs** . These knobs provide high-level control over multiple underlying parameters simultaneously, allowing you to quickly sculpt the overall sound and feel of the generated patterns. Examples of macro knobs include Pitch, Tone, Range, Noise, Density, Modulation, Intensity, and Space. The specific parameters controlled by the macro knobs can vary depending on the active engine.

The Macro knobs are your first level of direct, deliberate control of Scapeshift’s various engines, and they are in fact very powerful tools at your disposal. Experienced Scapeshift users often start here to broadly

shape their sounds and patterns before diving deeper into individual parameter controls. Experimenting with them is highly encouraged; more detailed descriptions of them can be found below.

Pattern Palette

When you generate a pattern you like, you can save it to the **pattern palette** located in the upper section of the interface. Clicking the large “**+**” button will add the currently active pattern to the palette for later use in sequencing or for Wander Mode.

Pattern Sequencer

You can add Patterns from your Pattern Palette to the Pattern Sequencer, which will morph from one pattern to the next over the period of time specified. It is worth noting that since the Pattern Sequencer is using saved, static Patterns, it will morph consistently between them and will play back the exact same thing every time. The implications of that last sentence are far more profound than one might think, as Chapter 7 will explain below.

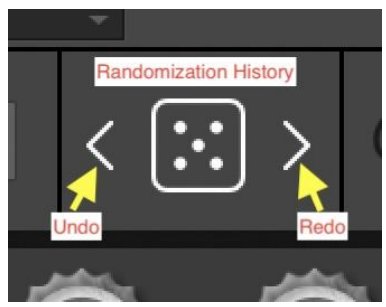
3. Generating Bass lines (Example)

The Bass engine in Scapeshift is a good example of how you can generate and manipulate musical ideas, so we'll start our deep dive into the various Scapeshift synthesizer engines here.

Single-Click Bass line Generation

The simplest way to create a bass line is to click the **dice icon** on the Bass engine Tab (to the left of the Theme Cards).

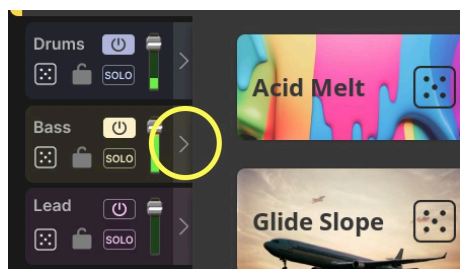
Each click of the dice will randomly generate a new (and often surprising) bass patch and pattern, potentially including note events and modulation curves that add expressive changes in timbre.



The combination of note events and modulation curves allows for the creation of bass lines that are expressive from the start. You can keep clicking the dice icon to explore a wide range of possibilities, from minimal to heavily syncopated and modulated bass lines.

Scapeshift conveniently keeps a history of all the dice randomizations you've generated, accessible through the navigation bar. This allows you to easily go back and revisit any interesting patterns you might have stumbled upon.

Sculpting with the Macro Knobs



After generating a bass line, you can use the Macro knobs to further shape its characteristics. If you are still seeing the Theme Cards rather than the Bass engine, you may need to click the Right Arrow on the Bass engine tab first:

Then you should see the Bass engine's Macro knobs at the top, which have a distinctive orange-brown colour. These knobs offer high-level control of just the Bass engine over several parameters simultaneously.

The available Macro controls include:

- **Pitch:** Adjusts the fundamental pitch of the pattern, i.e. the lowest note of the note Range (see below).
- **Tone:** Adjusts the filter cutoff frequency, making the sound brighter or duller.
- **Range:** Controls the overall pitch range of the bass line, from wide melodic leaps to almost single-note patterns.
- **Noise:** Adds or reduces the amount of noise in the sound.
- **Soft:** Controls the Attack settings of the various envelopes; turn up for longer attack times.
- **Dense:** Adjusts both the Decay and Release settings of the envelopes; turn up for longer decay and release times.
- **Modulation:** Controls the depth or strength of various modulations.
- **Intensity:** Controls the Cadence of the generated notes, measured in 16th notes between triggers; turning down decreases # of 16ths, resulting in faster Cadence/more triggers.

- **Space:** Adjusts both the reverb and delay sends.

By tweaking these Macro knobs, you can quickly tailor the generated bass line to better fit your needs.

If you find a particular pattern that you like, you can save it to the **pattern palette**. This allows you to build a collection of interesting bass line ideas that you can later use in your sequences.

Exploring Different Bass Styles

For a more directed approach to bass line generation, you can delve into the bass engine itself. Here, you'll find five different **style cards**, each representing a distinct flavor of bass line.

- **Subby:** Generates sub-bass focused lines without much high-end content
- **Walking:** Creates more melodic and "walking" style bass lines
- **Funky:** Produces syncopated patterns with more pronounced modulation and LFO use
- **Acid:** Generates classic acid-style bass lines
- **Gnarly:** Creates heavier, often dubstep-inspired bass lines

You can click on these style cards multiple times to get new variations within that specific style.

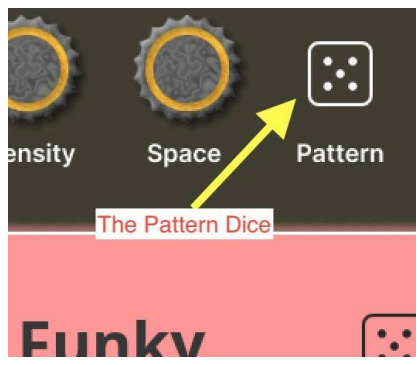
Additionally, you can blend between these styles by starting with one style, then clicking and dragging the dice icon of another style card before releasing. The more you drag before releasing, the more influence the second style will have on the resulting hybrid pattern.

This allows for creating hybrid bass line styles that sit between the defined style card categories.

Refining Patterns with the Pattern Dice

Once you have a bass line sound you like, the **Pattern dice** within the Bass engine–distinct from the Style dice–allows you to introduce subtle variations to the existing Pattern by clicking and dragging. Or, if you want a completely new Pattern, click without dragging for a fully randomized new Pattern.

These dice will only modify the pattern, leaving the sound parameters untouched. The resulting variations are also recorded in the randomization history.

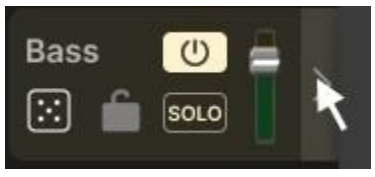


Bass engine Edit mode

In addition to the five Style Cards on the Bass engine page, you will see an “Edit” card as well. If you click on it, you will be taken to the page where you have control of every single parameter for both the Pattern and Sound of the Bass engine. Seasoned synth players will recognize many of these controls, but even a grizzled vet will be initially confused by some of them. We will go into the details a bit later, but if you want to take complete control over everything yourself and ignore the Dice, here’s where you do it!

Edit Mode Shortcut

If you click on the Bass engine tab arrow more than once, it will toggle between the Bass engine’s Theme Cards and the Edit Mode page. This handy little shortcut can save you from having to mouse over to the “Edit” card all the time, and it works on all of the other engine tabs as well.



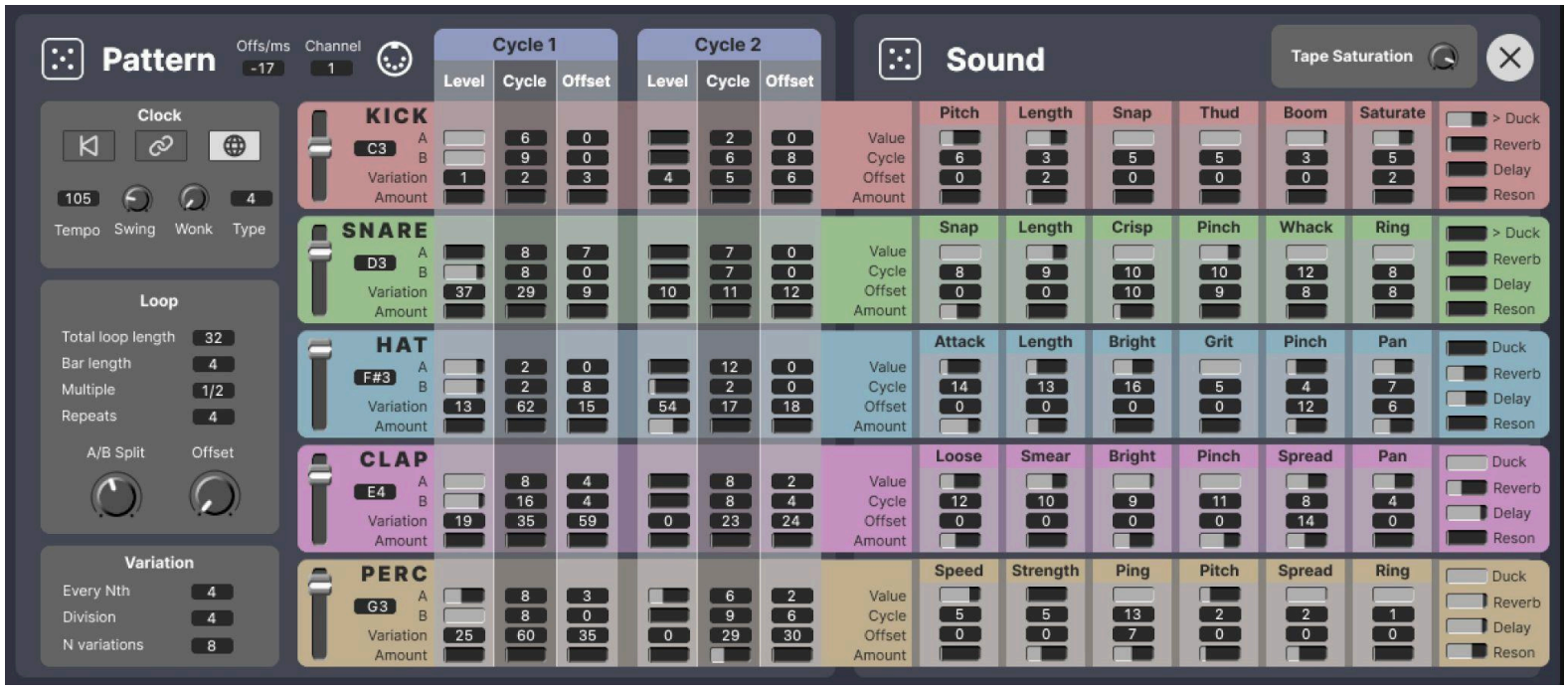
4. Exploring The Other Engines

Scapeshift features several other engines that can generate different musical elements to create a complete track. In addition to the Bass engine that we've just seen are the following.

Drums

Generate rhythmic patterns using a multitimbral drum synthesizer.

The Drum engine includes four Style Cards, a handy sub-mixer for the five "voices" of the drum synth, and an Edit page that is truly unlike anything else in Scapeshift.



Lead

Create monophonic melodic lead lines.

You may notice that the Edit window for the Lead engine looks suspiciously like the Bass engine. That is because the two are virtually identical mono synths.

Backing

Backing produces harmonic or accompanying parts. There is a bit more to this than meets the eye, though. Note the three tabs (highlighted in yellow here) just above the familiar Macro knobs! These tabs provide individual mutes and volume levels for the three respective backing engines: **Polysynth, Resonator** and **Noise.**

Polysynth

The Polysynth is a polyphonic FM synthesizer engine.

The Style Cards for the Polysynth are pictured immediately above, so we won't repeat them here. But here's the Edit page. You'll notice it's simply (FM? simple?) a two oscillator FM synth with some novel twists.

The Polysynth can trigger multiple notes simultaneously (up to around eight), with adjustable spread. Like other engines, pitch can be manually controlled and modulated by the pattern generator. The cadence of the Polysynth patterns can also be adjusted.

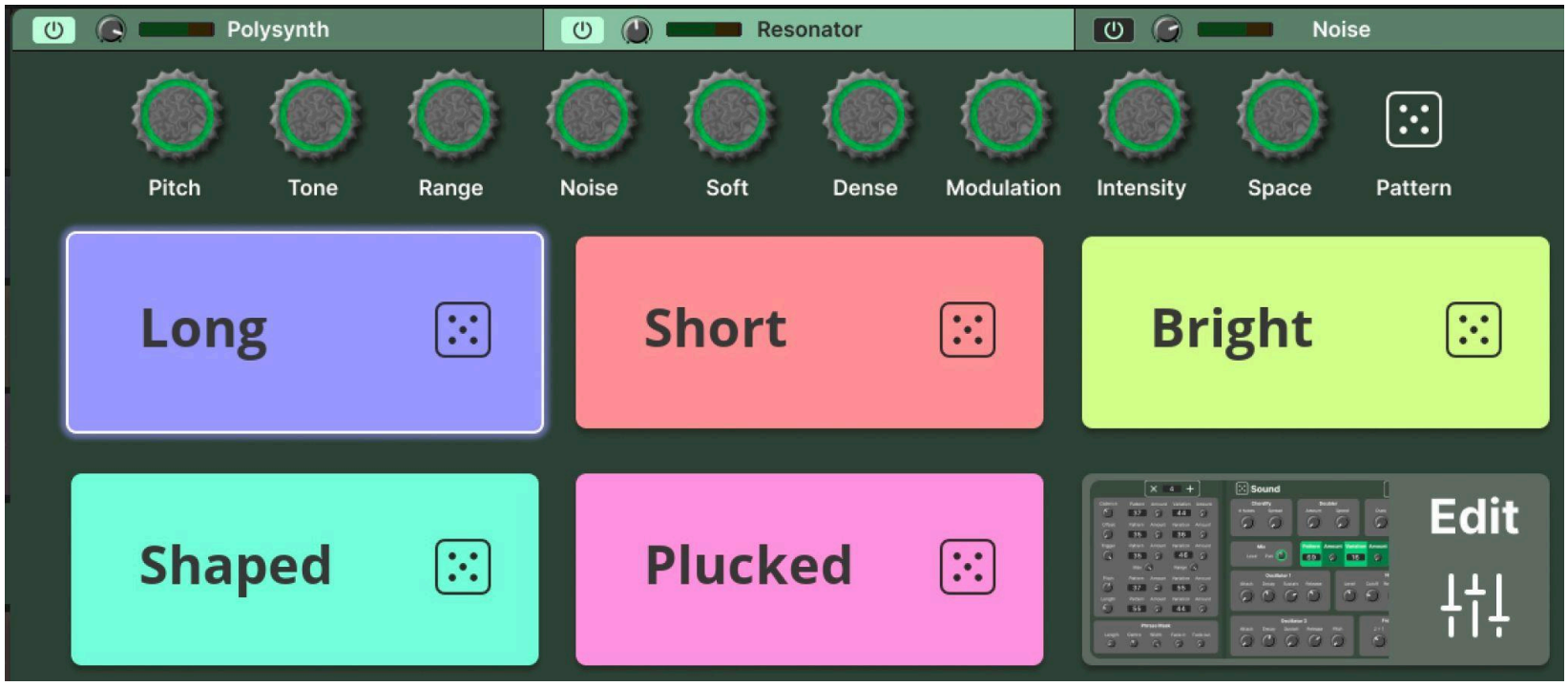
The architecture is based on a two-oscillator FM synth with an additional noise generator. The noise generator can also act as an FM source, and there is feedback FM between the two oscillators, enabling chaotic sounds. The number of notes played and their spread can be controlled. The Polysynth also features amplitude envelope controls for each oscillator and the noise generator.

Oscillator 2's pitch can be continuously changed relative to Oscillator 1 via pattern modulation, creating inharmonic effects. The FM section allows for **2 > 1** modulation (Osc 2 modulating Osc 1) and **1 > 2** modulation (Osc 1 modulating Osc 2), each also with pattern modulation capabilities.

The noise generator's level can be adjusted, and it has its own resonant bandpass filter with pattern modulation. Noise FM allows for **Noise > 1** modulation (Noise modulating Osc 1).

Resonator

This is a special module that’s half effect, half synthesizer engine. As such, it’s quite its own beast which comes with a full complement of Style Cards to go with it if you click on the Resonator tab above the Macro knobs.



Here’s the Edit page for the Resonator. Note that it has a spartan yet powerful Pattern section that allows modulation of the Resonator’s Base and Spread functions of the Chord, along with a Phrase Mask for shaping it:

The resonator is an effect consisting of four tuned comb filters followed by a modulatable low-pass filter that processes audio sent to it from other engines.

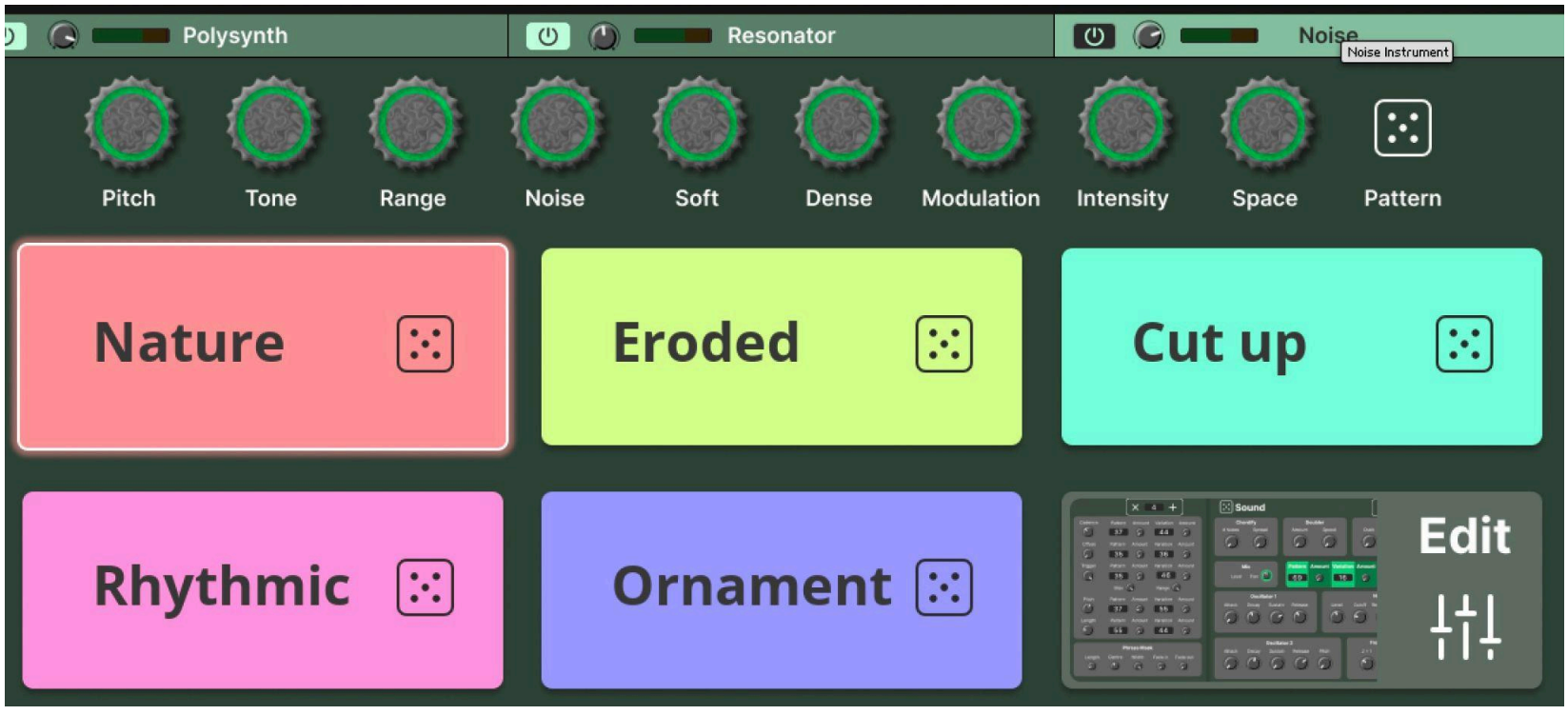
The resonator has parameters related to a chord: Base pitch, Spread of the notes, and a Cycle Length that can be used to modulate the base and spread, thus creating moving chords.

Resonator-specific parameters include Damp (brightness), Feedback (length and density), High Pass (cuts low frequencies), and FM (frequency modulation of the comb filters at their pitch).

The low-pass filter has cutoff and resonance controls, as well as envelopes that can be triggered by the drum triggers. The kick drum, for example, can negatively or positively trigger the filter envelope. Separate filter envelope attack and decay controls are also available. The resonator receives audio via pre-fader sends from the other engines, allowing for effects even without the original signal.

Noise

A sample-based noise engine. Like the Resonator, it has five Style Cards to go with it if you click on the Noise tab above the Macro knobs.



If you click on the Edit card to enter the Edit page, you'll see something unusual for Scapeshift in the lower right corner: **Sample**.

Samples in general don't fit in with what Scapeshift is doing with its parametric design and pattern morphing. But in this context, they're used more like complex oscillators and they work quite well. To be more specific, the Noise generator is a granular engine utilizing sampled field recordings that typically operates on an eighth-note pattern. It has envelope controls (Attack, Decay, Sustain and Release), all of which can be modulated by the pattern generator (see 6.3 below).

Key parameters include Sample Start Point (with pattern modulation) and Sample Select, allowing you to choose different field recordings for the Noise engine. In true Scapeshift fashion, yes: Sample Select can be pattern modulated, effectively sweeping across a range of the included samples. The pitch of the granular noise can also be modulated.

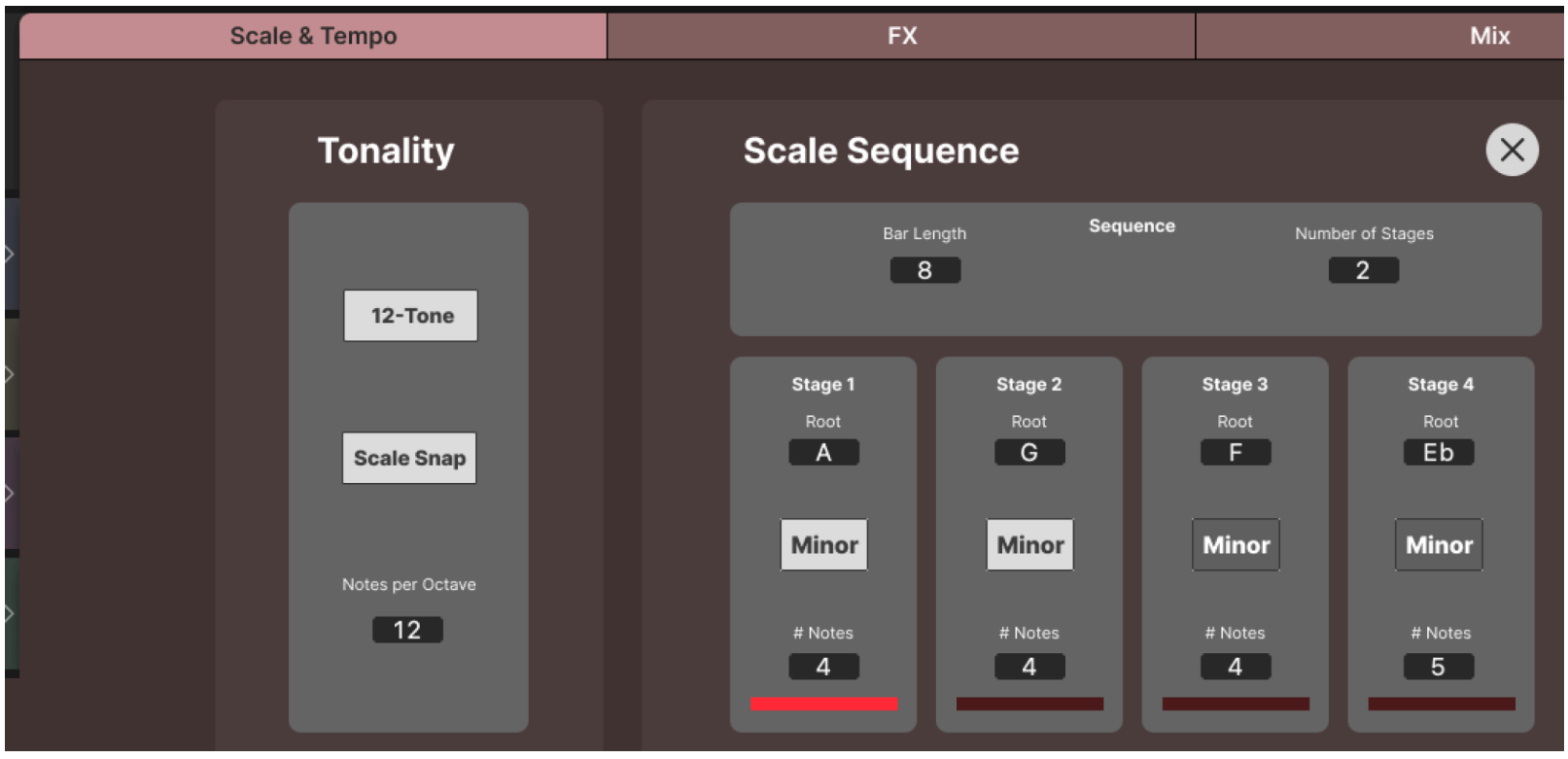
The Noise generator has a gentle filter used primarily for constraining the frequency spectrum, with high-pass functionality for creating hi-hat-like patterns if you wish. It also features saturator and bias effects for adding crunch. Like the other engines, it has a Resonator Send in its FX section for routing its output to the Resonator.

FX+Key

Generates sound effects and textures; controls global Pitch/Key/Tuning. Much like the Backing engine covered above, this is actually a tabbed collection of different but related things. Unlike the Backing engine, though, these controls are all Global.

Scale & Tempo

This crucial page/tab is a bit of a misnomer, since it doesn't really have anything to do with Tempo. It *does* have to do entirely with Scale & Tuning though! Click on the Scale & Tempo tab, and you'll see Tuning ("Tonality") controls on the left and "Scale" controls on the right. Use up to 4 stages in the Scale Sequence to set up chord progressions.



IMPORTANT TIP: While this whole section is Global, remember that it's Global *per Pattern* and can ***change from Pattern to Pattern!*** If you carefully select your Scale Sequence settings to build saved Patterns in your Pattern Palette, you can subsequently use the Pattern Sequencer to build deliberate and potentially complex chord progressions with them (See Chapter 7).

Another Tip: Most users will want to keep the "Tonality" settings at their defaults, as pictured above. However, if you unclick the 12-Tone and/or Scale Snap buttons, you can easily experiment with the weird

world of microtuning. Again, these global settings are saved and changeable *per Pattern*. So you can always build a microtonal pattern or two and save them in your Pattern Palette for a lysergic bridge if necessary!

FX

Scapeshift also features **effects** that can be applied to the generated sounds. They are integrated into Scapeshift's signal path and can be influenced by its generative processes.

There is also a **multi-band mixing** and **compression** system operating in the background.

Controls for Reverb and Delays

To the left, you'll see controls for the Global Reverb unit. And to the right, you'll see controls for both the Long Delay and Short Delay.

The **Reverb** is an effect that receives mix sends from the synth engines that are pre-fader. Parameters include High Damping (reduces high frequencies), Low Damping (can lead to a very dense sound at low settings), Size (from small/fast to large/slow), and Diffusion (at one end acts more like a delay with a grainy digital sound due to the interpolation of diffuser delays).

The **Delay** is also an effect that receives mix sends from the synth engines that are pre-fader. It features two delays: a long delay and a short delay, along with a filter that is a combination of a resonant high-pass and a resonant low-pass filter. In practice, this functions similarly to a dub delay.

Parameters include delay times for the long and short delays, filter cutoff and resonance (with resonance increasing feedback), filter spread between left and right channels, and an envelope follower on the filter. There is also a wobble control to simulate a tape delay effect. The short delay can be diffused to create flanger or chorus-like effects. Positive feedback can be achieved, potentially leading to howling sounds.

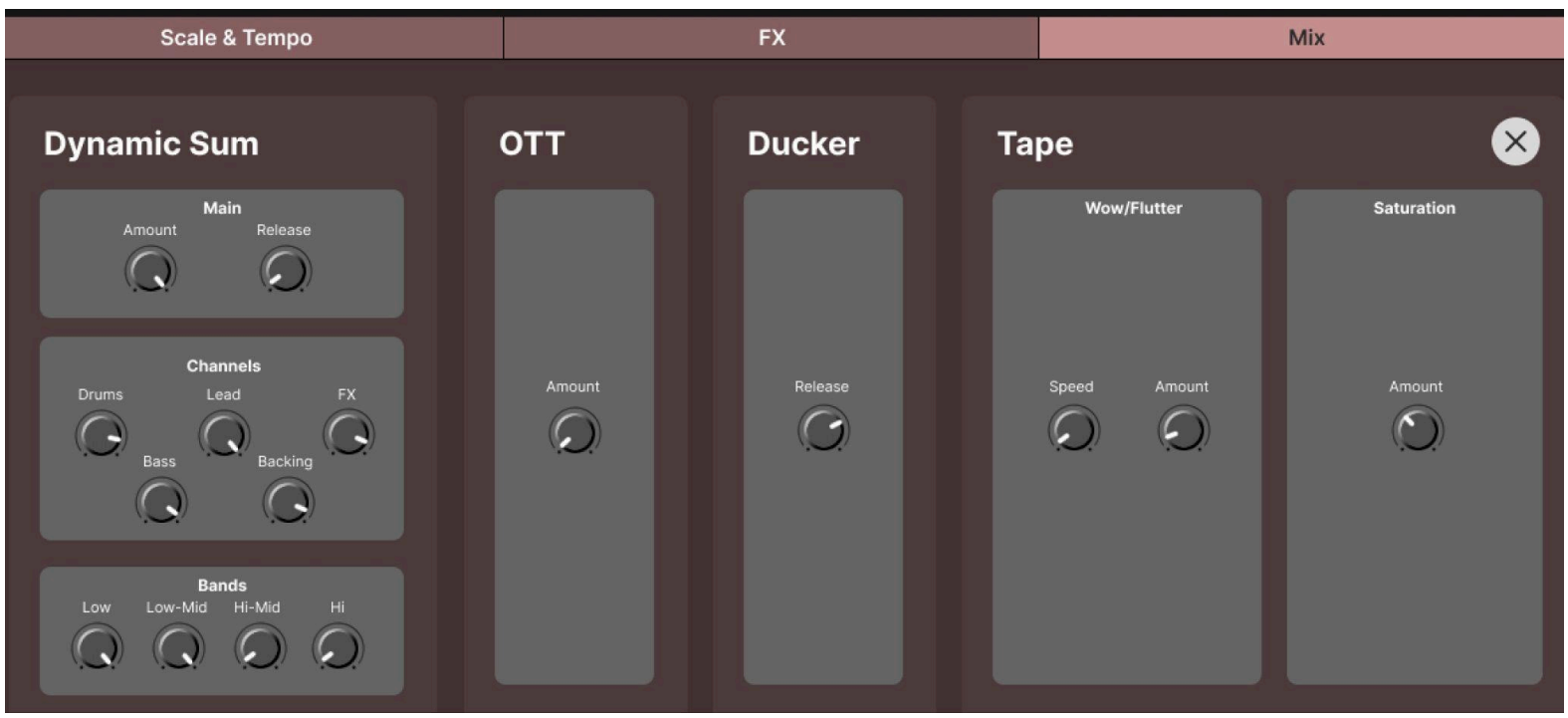
And since this is Scapeshift, there's a further twist! Note the Mixer Mode icon in the upper right corner of the Reverb section:

Click the icon to enter Mixer mode. When activated, the Stereo signal from external inputs 13 & 14 overrides the signal from the Reverb and Delay processors. This is useful if

you want to use the Scapeshift engine to mix stems recorded into your DAW from Scapeshift, or as a generic mix engine for anything coming from your DAW.

Mix

Scapeshift includes a built-in **mixer** that allows you to control the levels of the different engines. The mute and fader controls for each engine are part of the pattern morph data, meaning their states will change smoothly during transitions in the pattern sequencer. Note that soloing an engine is intended primarily for auditioning purposes and is not saved as part of the morph data.



On the left of the Mixer page is **Dynamic Summing**, which helps to carve out space for different elements in the mix, preventing a muddy final sound. The overall amount and the amount per frequency band (low, low-mid, high-mid, and high) can be adjusted. The amount applied to each bus (e.g., drums) can also be set, often with a lower amount for drums to help them punch through.

OTT Multiband Limiter: Adds presence to the overall sound.

Ducker: The release time of the Duck circuit can be adjusted here.

You can send varying amounts of the Drum engine's Kick and Snare audio to trigger the duck circuit using their **“> Duck”** controls. This creates an inverted envelope follower, and the **Release** knob found here controls its release time. There are individual "duck" controls on the other Drum tracks and Synth engines which control how much of their audio is ducked by this inverted envelope.

Tape Effects (Wow/Flutter): Simulates analog tape wow and flutter, from subtle pitch modulation at low speed to extreme warbling at high speed, adding warmth.

Tape Saturation: Adds warmth at lower settings and can be increased to introduce distortion and shredding.

For a more in-depth explanation of the various stages of the mastering chain, please refer to Chapter 11.

5. Outputting Audio and MIDI

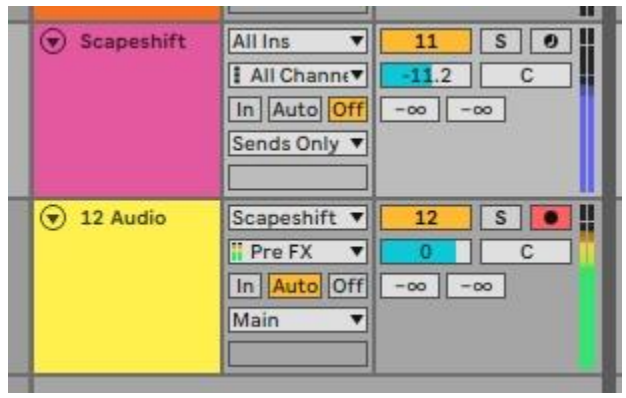
Scapeshift offers several ways to get your generated music out of the instrument.

Recording Audio Within Scapeshift

The simplest method is to use Reaktor's built-in **tape recorder** . Click the tape icon in the header to access recording and playback controls.

Arm the recording machine by hitting record, then start playback in Scapeshift to capture the audio output as a file. This audio file can then be imported into your Digital Audio Workstation (DAW) for further editing and arrangement.

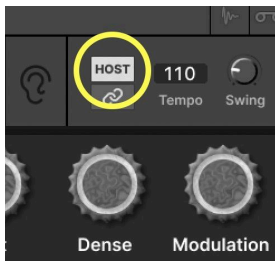
Recording Audio Directly into Your DAW



You can also record Scapeshift's audio output directly into your DAW. Load Reaktor as a plugin within your DAW, open Scapeshift, and set your DAW to record the audio from the plugin's output.

While every DAW is different, here's an example using Ableton Live.

Reaktor/Scapeshift is on Track 11, a MIDI track. The output is going to Sends Only and Monitoring is off, so you won't be able to hear this track directly. Meanwhile, Track 12 is an Audio track that is armed for recording. Monitoring is on (Auto). The audio source that it's using as input is the output of Track 11, Scapeshift. Healthy meters show the Scapeshift master outs are both audible and ready for recording.

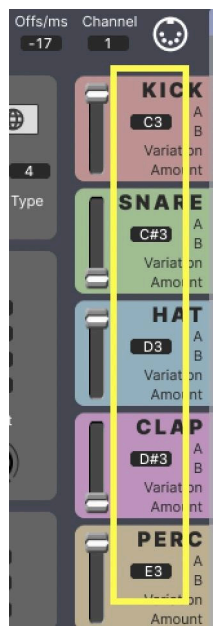


For synchronized recording, ensure **Host Sync** is enabled in Scapeshift, which will lock its playback to your DAW's transport and tempo.

MIDI Out

Scapeshift can send MIDI data from its generative engines to external hardware synthesizers, drum machines, or software plugins in your DAW. It can do this on an engine-by engine basis, so you could use Scapeshift synths for everything but the Bass track, which you may decide to send to a Minimoog VST3 plugin instead. In which case, good choice!

To enable MIDI output for the Bass engine, enter its edit view and click the **MIDI icon** .



When MIDI out is active, the internal sound generation for that engine will cease, and it will instead send MIDI notes and control change data. You can configure the MIDI channel for each engine. For the Drum engine, you can also define which MIDI note each of the five sequencing lanes will send.

“HELP! No Sound!”

If you find yourself saving your work, coming back later, and then can’t seem to make Scapeshift (or Reaktor) make any sound, *be sure to **check your MIDI settings** for the various engines!* Chances are good they were left turned on when you saved your work, causing all audio output from the engines themselves to be muted. Turn off the MIDI Mode for each engine to restore the audio outputs for them.

Triggering Software Plugins in Your DAW via MIDI

Most DAWs flatten the MIDI output of plugins to MIDI channel one, making multi-channel MIDI routing from Scapeshift difficult/impossible using only Reaktor and the DAW's built-in features.

To overcome this, you may well need to use a **wrapper plugin** that can host plugins itself and then output to virtual MIDI ports that your DAW won’t flatten to MIDI channel 1.

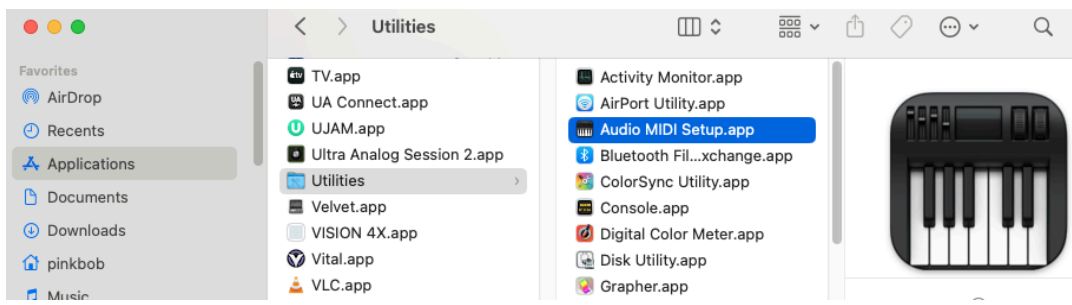
Examples of such digital modular plugins include **Kushview Element** (free) and **Plogue Bidule** (paid).

Setup process in a DAW (Windows)

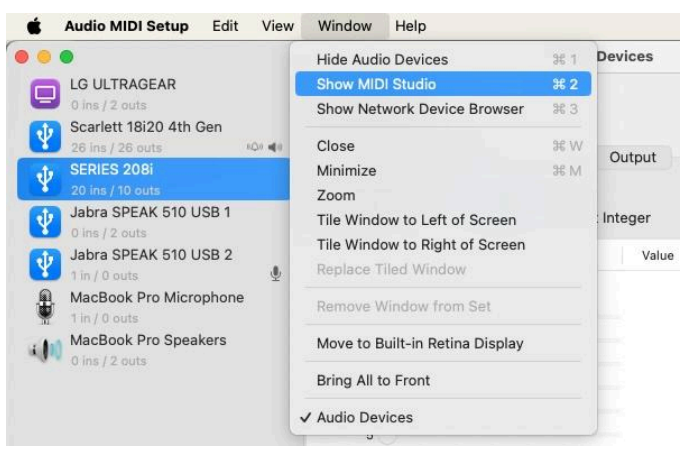
1. Download and install a third-party virtual MIDI port application like **Loop MIDI** .
2. Open your DAW and create a new MIDI channel.
3. Load either Element or Bidule as a plugin on this channel.
4. Within Element or Bidule, load Reaktor as a plugin.
5. Connect Reaktor's audio outputs to Element's or Bidule's audio output.
6. Wire up Reaktor's MIDI outputs to a virtual MIDI port within Element or Bidule.
7. In Reaktor, open Scapeshift and set the desired engines to MIDI mode, configuring their MIDI channels.
8. In your DAW's MIDI settings, enable the virtual MIDI port as a MIDI input device.
9. Create new MIDI tracks in your DAW and set their MIDI input to the virtual MIDI port, selecting the appropriate MIDI channels to receive data from Scapeshift's engines.

Setup process in a DAW (Mac)

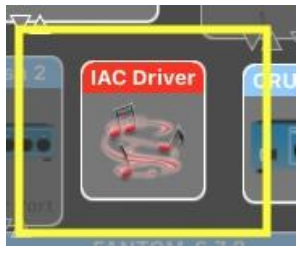
1. Open **Audio MIDI Setup** (Applications > Utilities).



2. Go to **MIDI Studio** (Window > Show MIDI Studio).



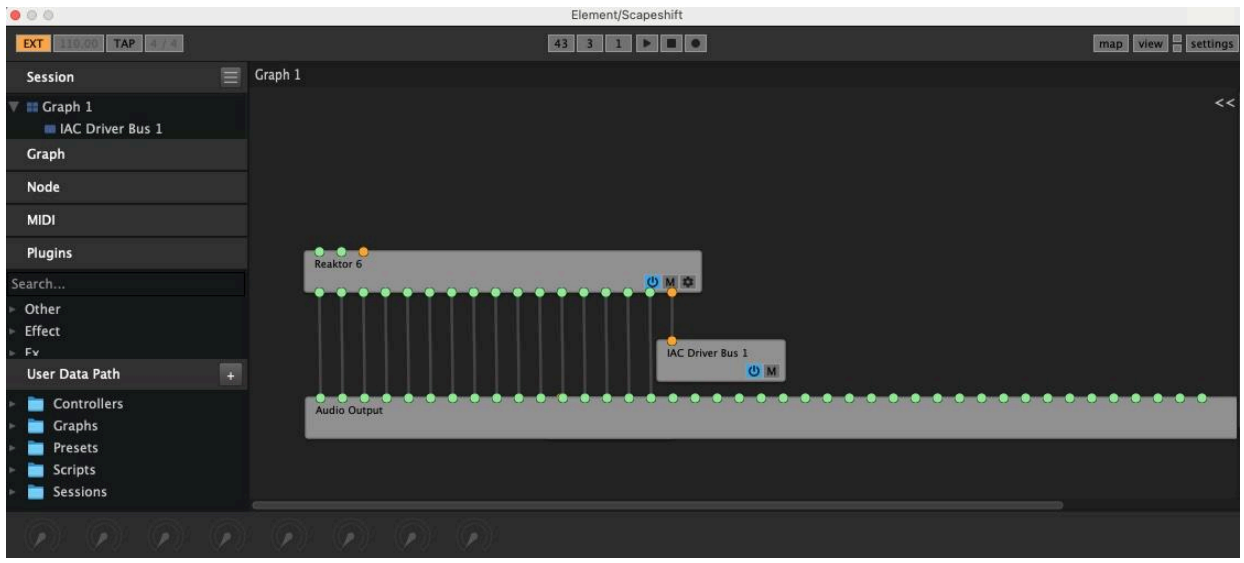
3. Double-click on the **IAC Driver**.



4. Ensure the "Device is online" checkbox is checked.

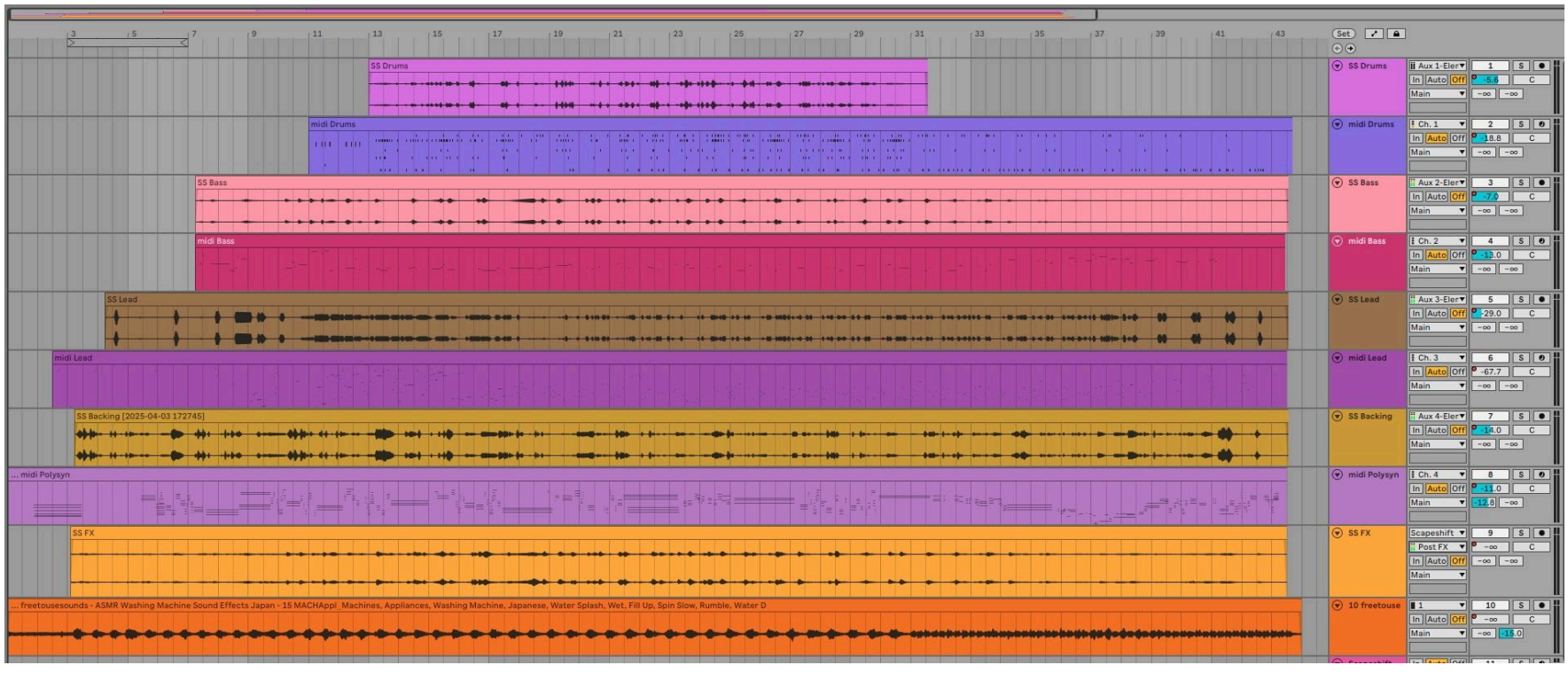
5. Follow steps 2-9 from the Windows setup steps above, using the IAC Driver as your virtual MIDI port.

Here is an example of what this wiring would look like on a Mac running Ableton Live using Element as the wrapper plugin. It is worth noting that on a Mac, this user reported issues with the multi outs failing to appear when Reaktor was loaded as an AU plugin, which is rather counterintuitive and may be system specific. So in this illustration, Reaktor was loaded as a VST3 plugin within Element instead, which solved the issue.



Multitrack Recording with Scapeshift in your DAW

Building on the Ableton Live/Element setup pictured immediately above, it is relatively easy to then record the audio and MIDI coming out of the various Scapeshift engines as discrete audio tracks. It is because of the wiring you see in the example above that the various engines will be available in your DAW as discrete stereo tracks (and MIDI tracks) suitable for multitrack recording. Here is a screenshot of a recording that is a good example of what you can build in a DAW, with Scapeshift and all of its generative capabilities at the heart of it all.



With the exception of the bottom dark orange track, everything here was generated by Scapeshift and then massaged and edited in Ableton Live. **And here is where the fact that the Pattern Sequencer reliably *does the same thing every time* comes into play.**

If you look closely, you'll see that besides the bottom orange track–which is actually a field recording of a washing machine in Tokyo because of course it is–there are five audio tracks and four MIDI tracks.

Four of the five audio tracks were recorded out of Scapeshift in a single pass. Capturing the FX audio track required a second pass that's detailed below, and then the four (initially five) MIDI tracks were recorded in a third pass with the MIDI Out option enabled on all sound engines. In this particular example, the Noise track ended up containing no useful MIDI information for the piece and was deleted, so only four of the possible five MIDI tracks remain in the screenshot.

Due to latency, the audio and MIDI tracks from the two passes had to be compensated for by zooming way in and manually shifting the MIDI tracks ever so slightly to align perfectly with the audio tracks. At this point, third party plugins could be added to each of the MIDI tracks in order to greatly enhance the sonic palette of Scapeshift's generated music. Finally, the Scapeshift synth engine audio tracks and MIDI-driven plugins were combined for the final mix.

In order to accomplish this, Reaktor 6 was loaded as a VST3 Plugin inside of Element within Ableton Live on a Mac, as pictured above. Routing was then set in Ableton Live to facilitate the multitrack recording of both audio and MIDI on successive passes.

Because the FX track *only* shows up on Reaktor's main outs, it had to be recorded on its own pass with everything but the FX/Key track muted within Scapeshift. This allows only the FX engine to appear at the main outs for this recording pass. This is important because the FX engine is where the Resonator appears. And due to the pre-fader send nature of Scapeshift, the FX track potentially contains the wet effected signals of the synth engines–even if they're normally muted (not just for this recording pass) in any given pattern.

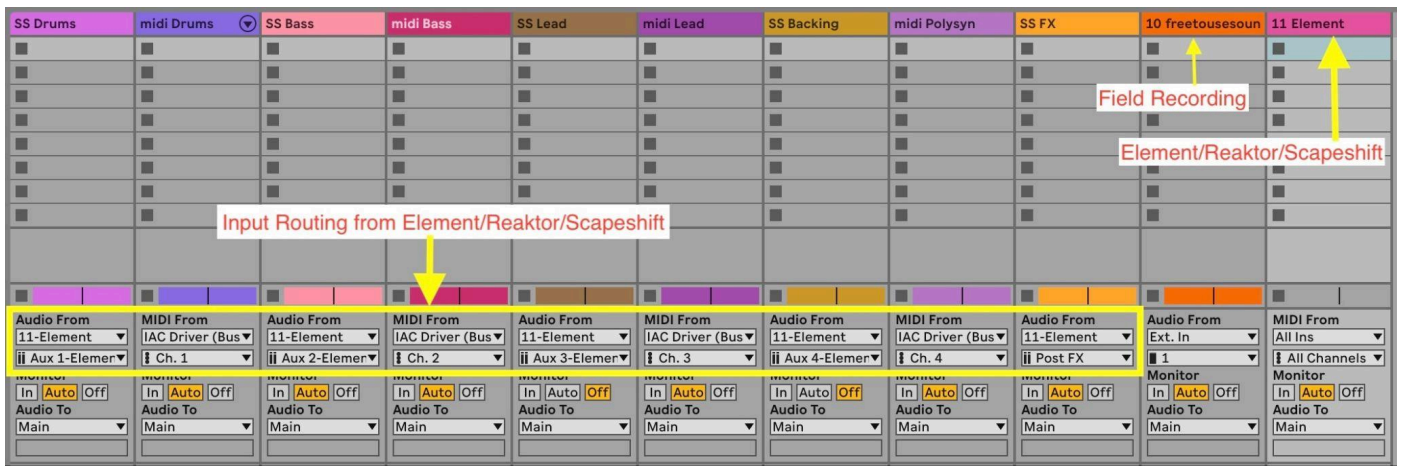
As stated at the start of this section, the initial setup of Reaktor with something like Bidule or Element is key to enabling this capability for recording both the audio and MIDI.

The default audio input routing in Ableton for the four Scapeshift instrument tracks in this example (excluding FX) is:

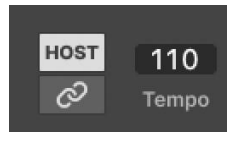
- Drums = Element Aux Out 1
- Bass = Element Aux Out 2
- Lead = Element Aux Out 3
- Backing = Element Aux Out 4.

For the MIDI tracks in this Mac-based example, the IAC Bus driver is used as the MIDI input source. Scapeshift's default synth engine MIDI channels are used for each of the five MIDI tracks:

- Drums=1
- Bass=2
- Lead=3
- Polysynth=4
- Noise=5 (missing in this example)



One final note: you'll need to have Scapeshift's HOST button enabled throughout the process so that everything is controlled by the DAW's clock/tempo and the multiple recording passes are all in sync with each other.



Triggering Software Plugins in a Standalone Digital Modular

You can also run Element or Bidule as standalone applications and host Reaktor and Scapeshift within them to send MIDI to other plugins hosted in the same environment. This setup allows you to route the audio output of these triggered plugins back into Reaktor to utilize Scapeshift's mix engine and effects.

Note that Element Standalone may have stability issues when loading Reaktor. Plogue Bidule is suggested as a more reliable option for this. Element may be more suitable for use inside of a DAW as outlined above.

Setup process in Plogue Bidule

1. Launch the Bidule standalone application.
2. Right-click in the main Bidule view and insert the **Reaktor 6 FX VST3** plugin (the FX part is important: this "effects" version supports multiple inputs).
3. Connect the first two left-hand audio output ports of Reaktor to the first two left-hand input ports of your sound card output device in Bidule.
4. Open Scapeshift within the Reaktor plugin. Set a pattern playing and enable MIDI mode for the desired engines, setting their channels.
5. In the main Bidule view, right-click and insert a **MIDI Splitter** object.
6. Connect the output port of the MIDI Splitter to Reaktor's MIDI input port.
7. Insert your desired plugin instrument in Bidule.
8. Connect the left output of the MIDI Channel Splitter (you may need to configure the splitter for the correct MIDI channel) to your plugin's MIDI input terminal.
9. Connect your plugin's first two audio outputs to the appropriate stereo input pair in Reaktor 6 FX. Scapeshift's audio inputs are mapped as follows:
   - Drums on stereo input pair 1,
   - Bass on stereo input pair 2,
   - Lead on stereo input pair 3,
   - Polysynth on stereo input pair 4,
   - Resonator on stereo pair 5,(only used if the Resonator "Mixer mode" button is enabled)

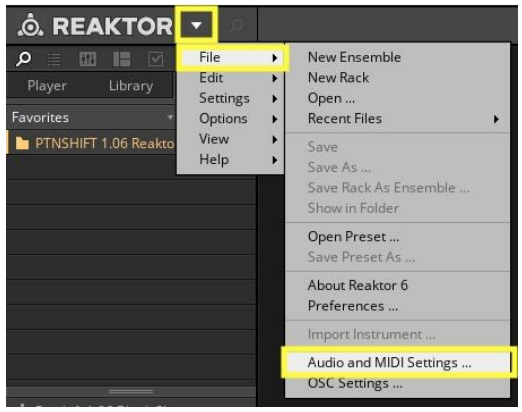
   - Noise on stereo input pair 6, and
   - FX return on stereo pair 7 (overrides the internal Reverb and Delay audio outputs with external audio, only if the Reverb "Mixer mode" is enabled).

Repeat this process for each Scapeshift engine you want to use to trigger external plugins.

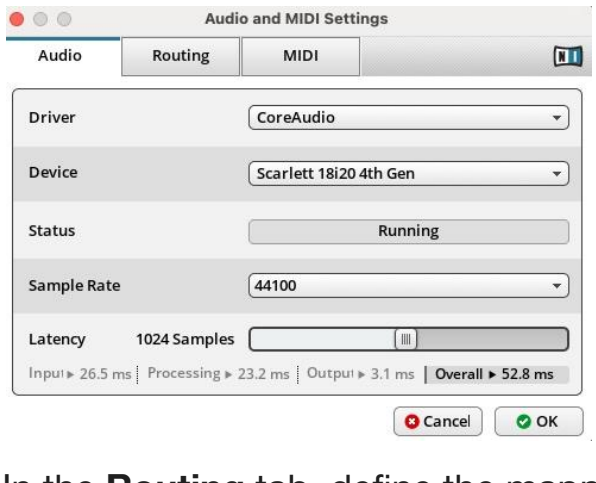
Triggering Hardware Synthesizers

For triggering hardware, it's recommended to run Reaktor in **Standalone mode**.

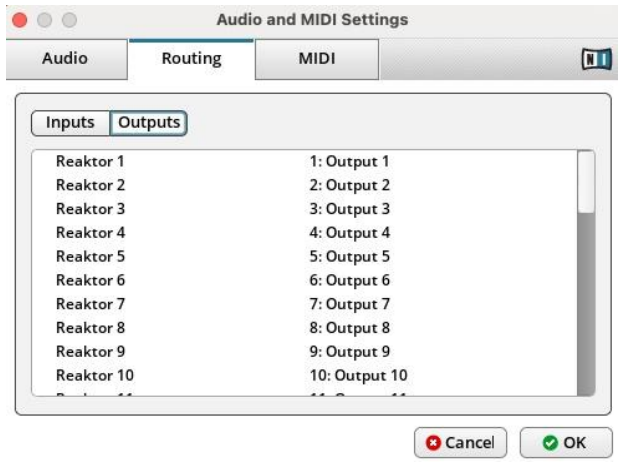
1. Launch the Reaktor application.
2. Go to File > Audio and MIDI Settings.



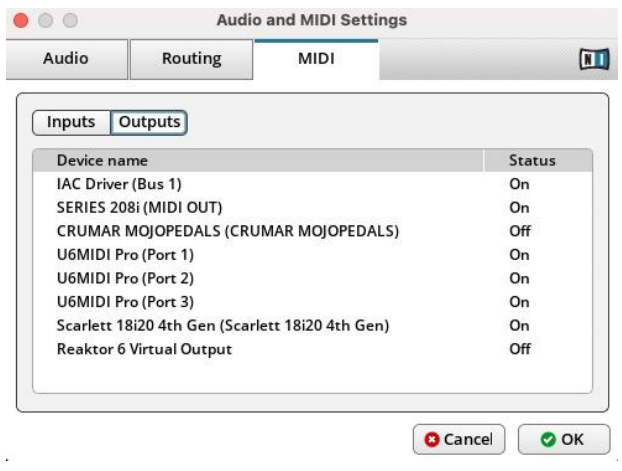
3. In the **Audio** tab, set up your sound card.



4. In the **Routing** tab, define the mapping between Reaktor's inputs/outputs and your sound card's inputs/outputs.



5. In the **MIDI** tab, enable the MIDI output ports that correspond to your hardware devices.

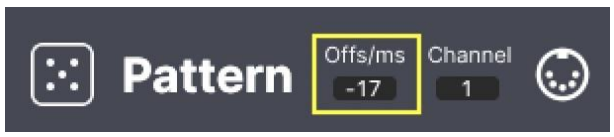


6. Open Scapeshift and enable MIDI mode for the desired engines, setting their MIDI channels.

7. Use your MIDI cables to connect your computer's MIDI output to your hardware synthesizers.
8. Route the audio output of your hardware back into your audio interface and, if desired, into Reaktor's audio inputs for processing with Scapeshift's effects.

Latency Compensation

When using MIDI out and routing audio back into Scapeshift, you may encounter latency issues. Scapeshift includes a **latency compensation** feature with an offset control for each engine. You can adjust these offset values (in milliseconds) to synchronize the timing of external instruments with Scapeshift's internal rhythm.



Audio Input

Scapeshift also features **audio input**, allowing you to route external audio signals into its signal path. When MIDI out is enabled for an engine, it can also override its internal audio signal with a live stereo input on specific channels in Reaktor. For the Bass engine, this input is typically on channels 3 and 4. This allows you to process the audio from external synthesizers (triggered by Scapeshift's MIDI out) through Scapeshift's multi-band summing, multi-band compression, and effects. The audio inputs are mapped as follows:

- Drums on stereo pair 1 (Ch 1-2)
- Bass on stereo pair 2 (Ch 3-4)
- Lead on stereo pair 3 (Ch 5-6)
- Polysynth on stereo pair 4 (Ch 7-8)
- Noise on stereo pair 6 (Ch 11-12)

6. Deep Dive into Pattern and Sound Controls

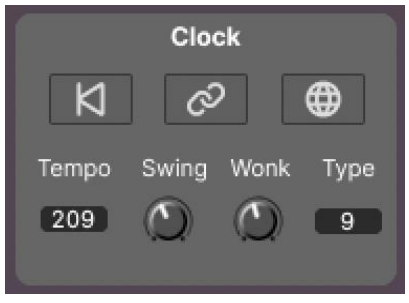
For a more detailed level of control, Scapeshift offers an Edit view for each engine, which we've already seen briefly in Chapters 3 and 4. The Edit view is divided into Pattern controls and Sound controls. Here's another look at the Lead engine's Edit page with the two clearly marked. Note that each has its own set of draggable dice icons for introducing randomness in one without the other.

Pattern Controls

The pattern section governs how musical events are generated. Key parameters include:

Clock

Controls the main tempo and synchronization. It can be synchronized to the host DAW or de-synced using the Global button.



Loop

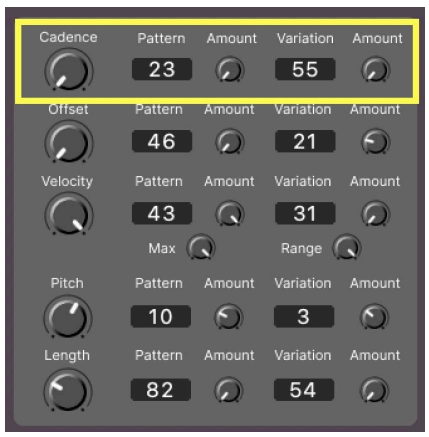
Determines the overall length of the pattern in bars and the subdivision of the bar (e.g., half measures, quarter measures). The total loop length is displayed in 16th notes. You can also set the number of repeats for a loop.

Note Controls

The next block of controls don’t have a label, most likely because there wasn’t any room for one! But if it had one, it would probably say “Note”. The following controls will help you shape the timing, pitch, density, volume and phrasing of the notes that Scapeshift generates.

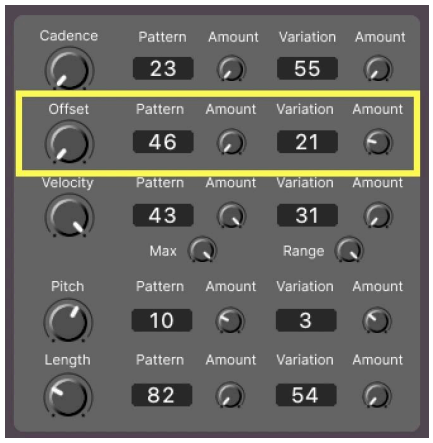
Cadence

Determines the time interval between each note event in the sequence. Increasing it makes the events more spread out. You can also apply modulation to the cadence over time, creating rhythmic variations.



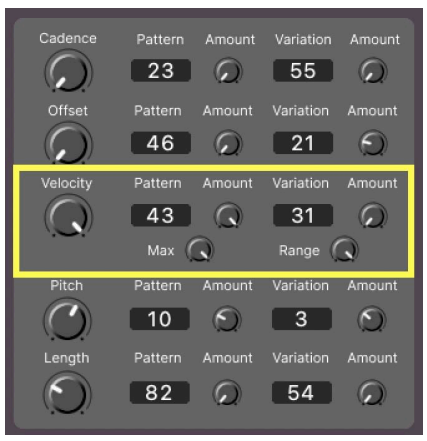
Offset

Shifts the timing of the note events within the 16th-note grid of the bar. Modulating the offset can create interesting off-beat rhythms.



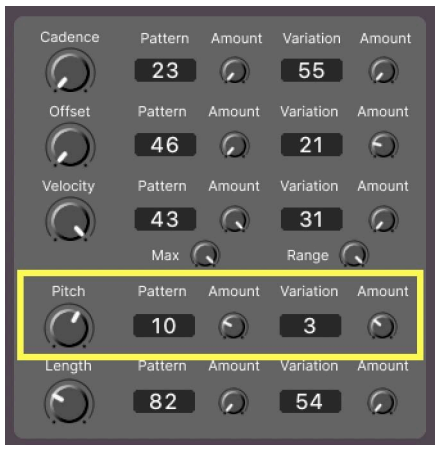
Velocity

Controls the attack strength of each note. Modulating velocity adds dynamic expression to the pattern. The velocity section also features controls to filter out notes based on their velocity range, offering another way to shape the pattern.



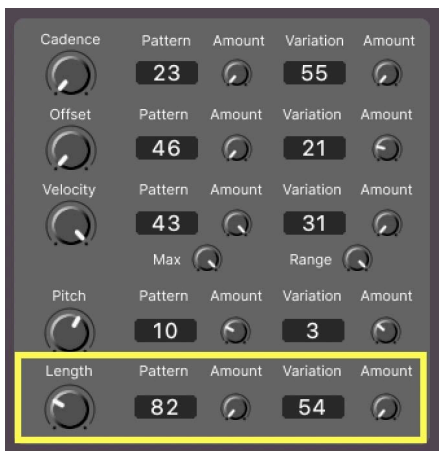
Pitch

Sets the fundamental pitch of the generated notes. Applying modulation to the pitch parameter creates melodies.



Length

Determines the duration of each note. Modulating the note length can create staccato, legato, or overlapping sounds.



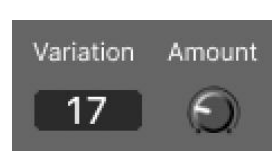
The above descriptions mention applying modulation. But since “modulation” usually implies “LFOs,” you may be wondering why you don’t see any LFOs here? Since Scapshift is different by design, it uses “Patterns” and “Variations” for modulation instead of LFOs. And in an attempt to clear up any initial confusion, *these patterns are **totally different** from the patterns that you can save in the Pattern Palette.* Here is a description of the modulation patterns:

Pattern and Amount Knobs

For each of the above parameters, you'll find a "Pattern" and an "Amount" knob.

The "Amount" knob controls the intensity of the variation applied to the parameter over time–just like the Amount control you’d find associated with an LFO on a conventional synth. However, the "Pattern" drag control selects a mathematical algorithm that generates this modulation, in place of a typical LFO’s boring “sine” or “sawtooth” patterns. Neighboring pattern numbers produce similar modulations, allowing for smooth transitions. Pattern numbers that are further away from each other will be more dissimilar.

Variation and Amount Knobs



This introduces subtle sub-variations to the main pattern at defined intervals (e.g., every 17th quarter note). This helps keep the pattern evolving without becoming repetitive. You can control the interval and the amount of variation applied.

Phrase Mask

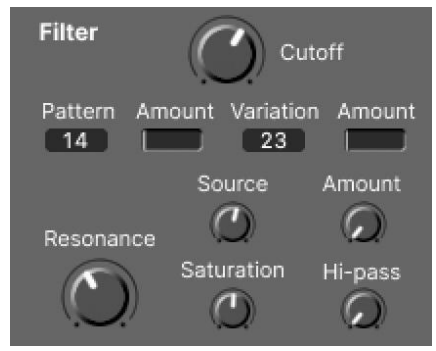
Patterns can be ‘masked’ after their generated output to selectively allow some events through and throttle others.

The Phrase Mask functions create an oscillating mask, oscillating at a **Length** which is a proportion of the overall length of the pattern. **Width** controls what percentage of the cycle events are allowed to pass through. At maximum Width, all events are passed; at minimum Width, no events are passed. The **Centre** of the pattern can be adjusted to provide an offset for the range of passed and throttled events. The **Fade In** and **Fade Out** parameters introduce a fade curve at the rising and falling boundaries of the phrase mask.

Monosynth Sound Controls

The sound section allows you to sculpt the sonic characteristics of each engine. The Bass and Lead engines, for example, are nearly identical dual oscillator monophonic synthesizers. Common controls include:

The Filter



Cutoff

Controls the cutoff frequency of a filter, shaping the tonal brightness of the sound. Unlike most Sound parameters in Scapeshift, Filter Cutoff is such an important parameter that it gets its own dedicated Pattern/Variation controls similar to those in the Pattern section of an Edit page (see above). As for parameters that don’t have their own dedicated controls, we’ll get to that in a moment in section 6.3.

Resonance

This is a standard resonance control that accentuates the frequencies around the cutoff frequency.

Source/Amount

These two knobs control “conventional” modulation of the filter. In addition to the Scapeshift way of adding modulation through pattern/variation controls, you can also use a filter envelope, the LFOs, or a blending of the two. **Source** is the source of the modulation: hard left is Filter Envelope only, and hard right is LFO section only. **Amount** is how much of the modulation is actually applied.

Saturation

Can be used to add some distortion and crunch to the filter circuit.

Hi-pass

The main filter we’ve been discussing here is of course a low pass filter. But what if the bass is getting *too* subby and you just want the pain to stop? This high pass filter at the end of the filter chain allows you to cut low end. Hard left is “Hi-pass filter off”; turn clockwise to remove more and more bottom end.

The Oscillators

Oscillator 1

You can select the waveform **Type**, ranging from sine to triangle to square to sawtooth, and add noise.

Oscillator 2

Similar to Oscillator 1, Oscillator 2 offers various waveforms with the **Type** knob.

Crucially though, oscillator 2 can **sync** to oscillator 1, creating a wide range of complex timbres depending on the **sync amount**.

Oscillator 2 also has its own built-in FM oscillator**;** you can control the **FM amount** and the **FM pitch** (the frequency of the modulating oscillator) to create rich, frequency-modulated sounds.

As with the Filter, the Source knobs here control the blend between envelope and LFOs, and the Amount knob controls the amount of modulation applied. For the Source setting, hard left is all Mod Envelope, and hard right is all LFO section.

The Envelopes

Scapeshift features three envelopes:

Amp Envelope

Standard ADSR (Attack, Decay, Sustain, Release) envelope that controls the overall volume contour of the sound.

Filter Envelope

Modulates the filter cutoff over time, allowing you to create dynamic sweeps and tonal shaping.

Modulation Envelope

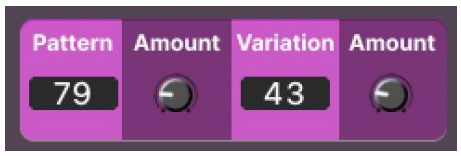
A general-purpose envelope that can be assigned to modulate various sound parameters.

The LFOs (Low-Frequency Oscillators)

There are two LFOs available, which can be mixed together. LFO 1 is tempo-synced, while LFO 2 is un-synced and can reach higher frequencies for more drastic tonal changes. You can control the speed and waveform of the LFOs.

Modulating Sound Parameters Over Time

For many of the sound parameters, you'll find a **Source** and an **Amount** control as mentioned above. But there's more than that. *WAY more!*



A powerful feature of Scapeshift is the ability to modulate almost any sound parameter over time using the same modulation pattern controls documented in the "Pattern Controls" section above. When you click on a sound parameter knob, a yellow highlight appears around it, indicating that it can be further modulated.

This indicates that you can set the Pattern/Variation controls in the Modulation box pictured above for that (and only that!) parameter.

For example, you can modulate **Filter Resonance** over time by clicking its knob and then dialing in an amount and selecting a pattern (and variation) for its modulation.

Similarly, you can modulate the **Pitch of Oscillator 2**, the **Attack** and **Decay** of the various envelopes, and even the pitch **Slide** time on a per-note basis, creating a "slippery" groove. This capability allows for incredibly expressive and evolving sounds that change and morph throughout the pattern.

NOTE: Some parameters – notably the FX Sends and Doubler controls – are not selectable/modulatable in this manner.

If you click on a knob and it doesn't highlight in yellow, you can't control it with the Pattern/Variation controls.

Polysynth Sound Controls

As mentioned in Chapter 4, the Polysynth is a polyphonic FM synthesizer engine. If you dive into the Edit mode of the Polysynth–you may have to click the "Backing" engine tab, then the "Polysynth" tab at the top, and then the "Edit" card in the lower right to get there–you'll see that the "Pattern" side of things is the same as the Monosynth engines. But the "Sound" side of things is quite different.

You can think of this as a very unusually particularly weird mini DX7.

Having invoked the name of the legend, this is no DX7, to be sure. For starters, it has two operators (fancy FM-speak for oscillators) vs. the DX7's six. And we won't get into the DX's expanded envelopes. But this little Polysynth has a few tricks up its sleeve.

This manual is not intended to be a programming guide for FM synthesis, which is a notoriously gnarly topic. It is a skill gained over time, and experimentation is key. It's easy to go over the top with some of the controls, but you will eventually get better and better with FM through experimentation. The ancient joke comes to mind: "Doctor, it hurts when I do this." Doctor: "Well stop doing that!" So with all that said, let's dive into the Polysynth parameters!

Oscillator 1

While one would expect to see "normal" oscillator controls here, remember this is FM. So the waveform is "sine," and the frequency is fixed. Instead, here you see a standard ADSR amplitude envelope for shaping the sound of the "carrier" in this synth, Osc 1.

Oscillator 2

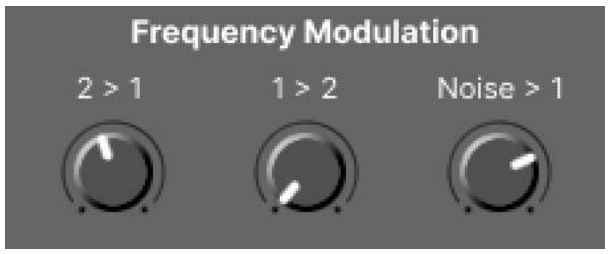
Osc 2's parameters are much like Osc 1: it's an ADSR amplitude envelope. But since Osc 2 is the "modulator" of Osc 1, it has a **Pitch** knob. As a modulator, you will never hear it directly. But if you have it modulating Osc 1, this Pitch knob will have a profound effect on the frequencies coming out of Osc 1.

At its highest setting, Osc 2 is at a 1:1 relationship with Osc 1, which–sonically speaking–is going to sound pretty sweet. Anywhere else on the dial, though, and you encounter the magical weirdness that only FM synthesis can bring to the table. Hint: modulate this (and other parameters) with the Pattern/Variation controls for shifting weirdness.

Noise

This is a noise generator specific to the Polysynth engine that has nothing to do with the sample-based "Noise engine" (see Chapter 4.3.3). While it can be used as a modulator for Oscillator 1, it can also be heard on its own by turning up the **Level** knob, which controls its output level. The **Cutoff** and **Resonance** knobs control a bandpass filter, while **Attack** and **Decay** control a simple yet useful amplitude envelope. With the level knob up, it's pretty much good for making cool hi hat sounds. With some modulation, it's pretty much good for making WAY cool hi hat sounds. But its primary purpose is as a second modulator. Speaking of which…

Frequency Modulation



Here is the central control of the "FM" part of the Polysynth. The **2 > 1** knob controls how much the modulator Osc 2 modulates the carrier, Osc 1. Similarly, the **Noise > 1** knob controls how much the Noise generator modulates carrier Osc 1. The **1 > 2** knob controls how much the Carrier (Osc 1) modulates the Modulator (Osc 2). Hmm. Yes, these can all be modulated individually via the Pattern/Variation box above them.

Chordify

This section is pretty simple. **# Notes** adjusts your polyphony from a few notes to about 8. **Spread** will adjust them from a range of one octave to…about 8!

Doubler

This is another simple set of controls, and something the Polysynth has in common with the two Monosynths. It's a pretty standard doubler effect you can use to thicken your sound. High **Amount** values coupled with low **Speed**–and vice versa–can be very interesting. As well as any point in between, for that matter.

Pitch LFO

These are standard Pitch LFO controls that can be used for thickening or special effects. A fairly low **Speed** setting is a good place to start, then turn up **Amount** to taste**.**

FX

Here are the pre-fader effects sends. They should be self-explanatory except for **Duck**, which controls how much of the signal is sent to the Ducker effect to potentially be ducked. It does *not* control how much this engine ducks other engines, as only the Kick and Snare (from the Drum engine) can do that.

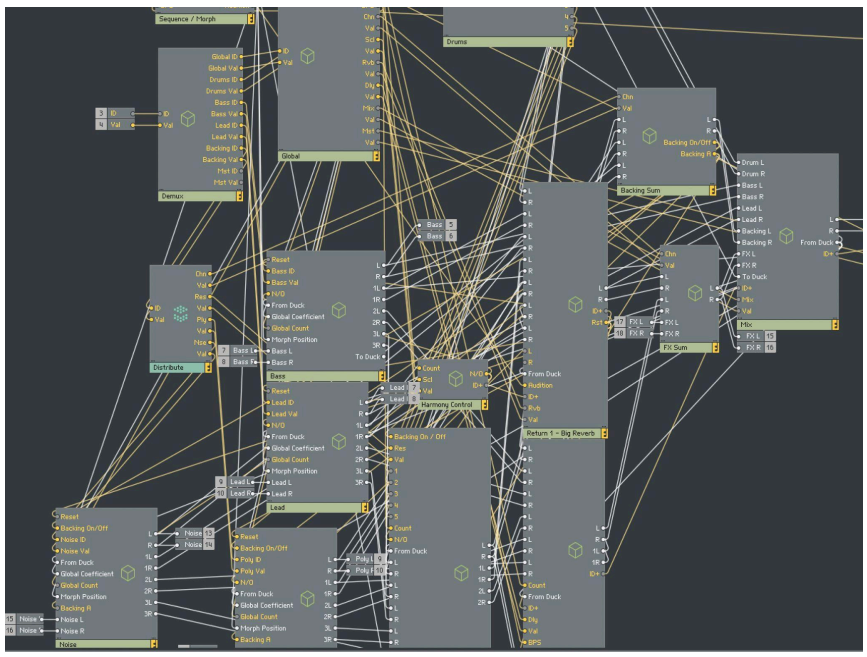
Mix

Finally is the Mix control, including the bonus phantom label! We will assume that the Level knob was removed during development for technical reasons.

"Technical reasons?"

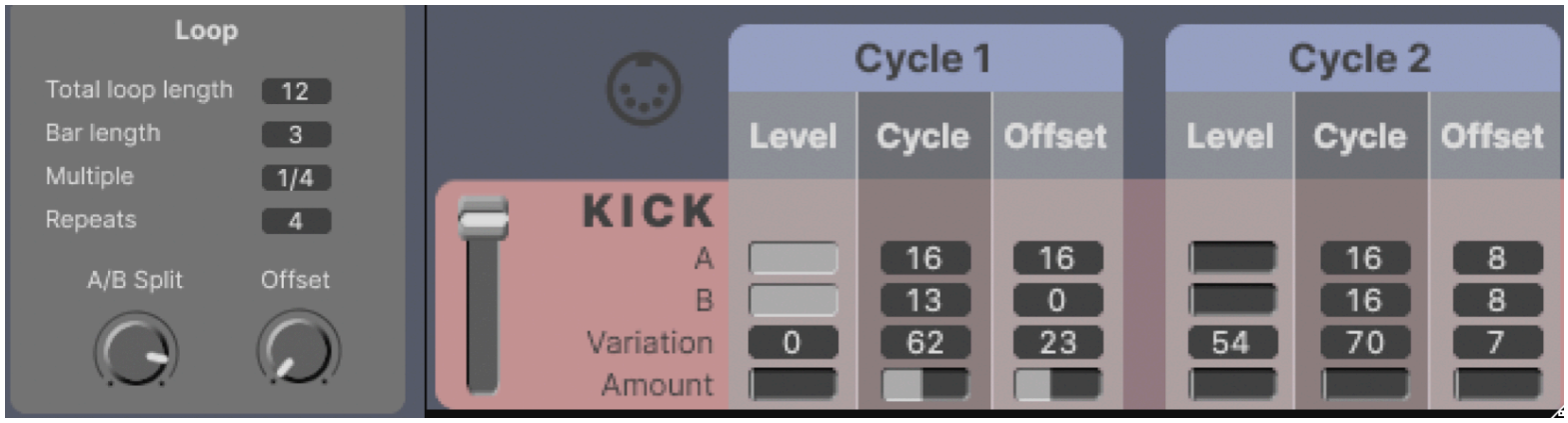
Yes. Have you ever contemplated what's actually going on inside Scapeshift? Since that Pan knob is self-explanatory–*and of course it can be modulated!*–here's a glimpse of just a tiny corner of it.

WARNING: viewing this image may cause vertigo!



Drums

The Drums engine utilizes a system of two independent **trigger-generating Cycles per voice**. It is important to note that these two Cycles run in parallel to each other **along the same timeline**–*not* one after the other successively. Don't worry, there is an illustration coming up that (hopefully) makes all of this clear. While complex, this unique Drum synth engine is incredibly powerful and worth taking the time to learn.

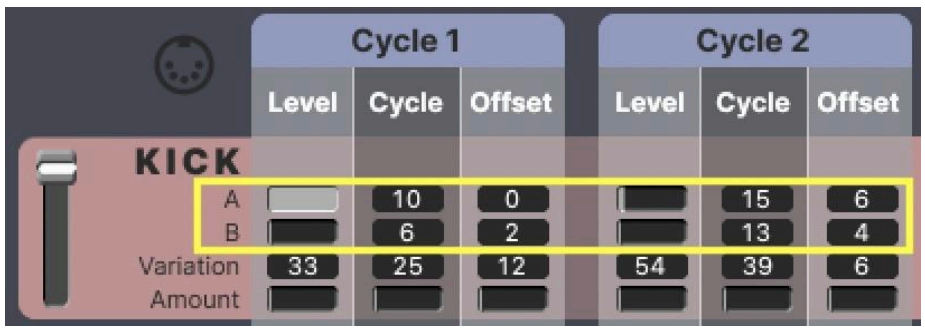


Pattern Controls

Cycle, Split and Offset

The **Cycle loop length** for all drum voices is set by the **Loop** section (the gray box pictured above). "**Bar length**" is measured as the number of notes of the duration specified by the **"Multiple"**, so in the example above, the bars are in ¾ time…a waltz! If you don't really want to work in ¾ time, a Bar length of 4 will set the drum voice cycles to a bar length of four quarter notes, or sixteen 1/16th notes. Multiple can be set to values of ⅛, ¼, ½, or 1, corresponding to eighth, quarter, half and whole notes.

Multiple and Bar Length combined gives rise to the **Total Loop Length** that the 'Variation' parameters operate on - see "Variation" below. The generated triggers for each pair of Cycles are combined and sent to the relevant drum voice engine:

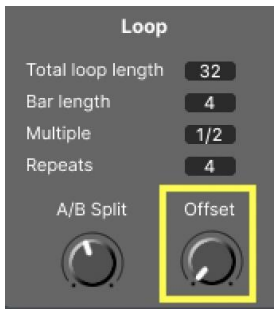


Each drum Cycle consists of two trigger-generating **Parts**, **“A”** and **“B”**.

Like the Cycles themselves, both “A” and “B” parts run in parallel, both starting from the start of each Cycle bar. However, only the triggers from one part are actually sent to the drum voice at any given time.

The **Loop** section’s “**A/B split**” knob controls when the “A’” and “B” part triggers are used. “B” always follows “A”, and the A/B split knob sets the point at which B takes over from A.

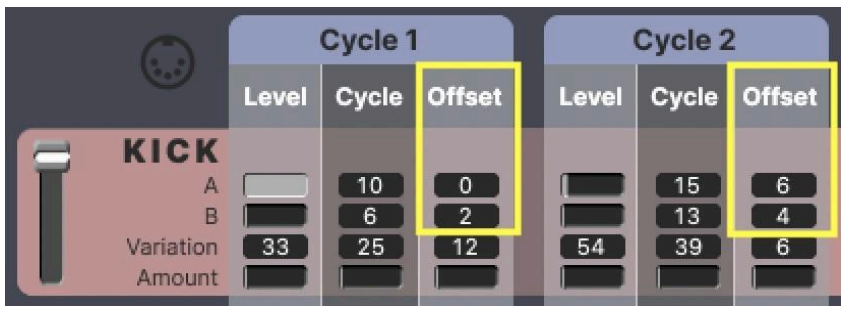
Turning the **A/B split** knob fully clockwise will send the “A” part’s triggers for the entire cycle loop. Dialing it counter-clockwise will introduce more of the “B” part, until when fully counter-clockwise, only the “B” part triggers will be used. The A/B Split point is effectively "picking up" the B pattern rather than "starting" it. This is a great live performative control for varying drum patterns. Also note that the A/B Split applies to all five drum voices globally; you cannot set different A/B split points per drum voice.



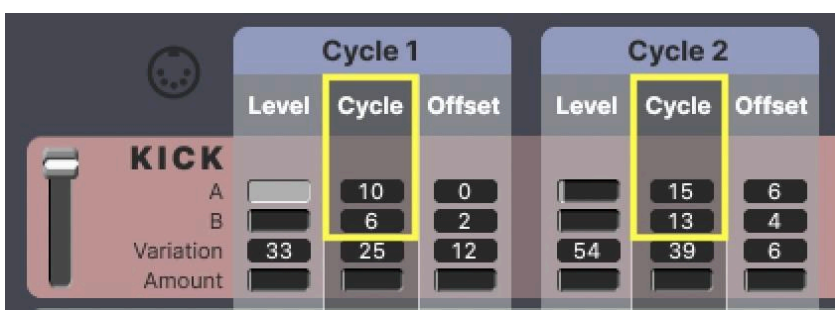
Each cycle and its parts “A” and “B” start from the beginning of each cycle loop. The start of the cycles and parts can be delayed using the Loop **“Offset”** knob, which is measured in 1/16th note units.

Fully counterclockwise corresponds to 0 sixteenth notes (i.e. no offset), while fully clockwise is equal to 16 sixteenth notes–a full 4/4 bar.

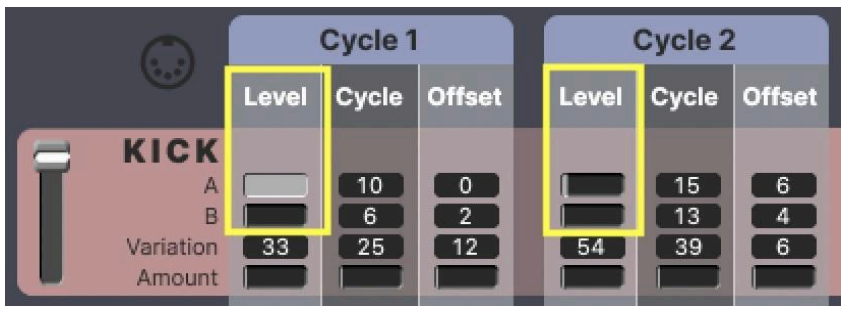
The first trigger of each cycle part will be generated after the cycle part’s “**Offset**” value, which is measured in 1/16th note units. Note that *this is different from and in addition to the Loop Offset* discussed above and can be set on a per-Cycle and per-Part (A and B) basis:



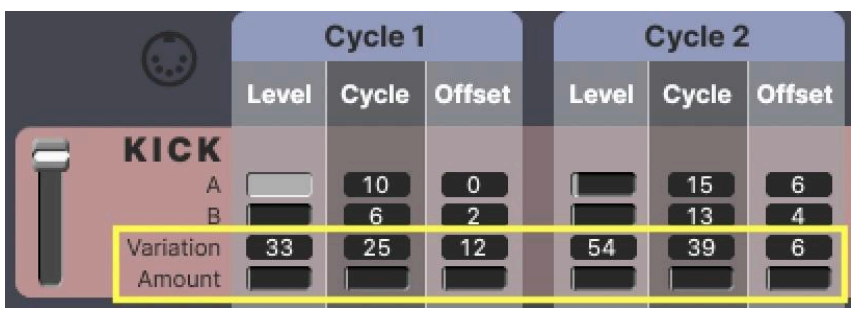
At the Offset value, a regular series of triggers are generated, one trigger every [Cycle value] x 1/16th notes:



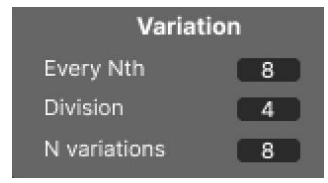
The volume or strength of each trigger is set by the Cycle part's "**Level**" control. Setting the Level to zero stops all the triggers from firing from that Cycle part:



The regular trigger sequence for each Cycle part can be varied between bars by dialing in the "**Variation**" control (0 - 100):

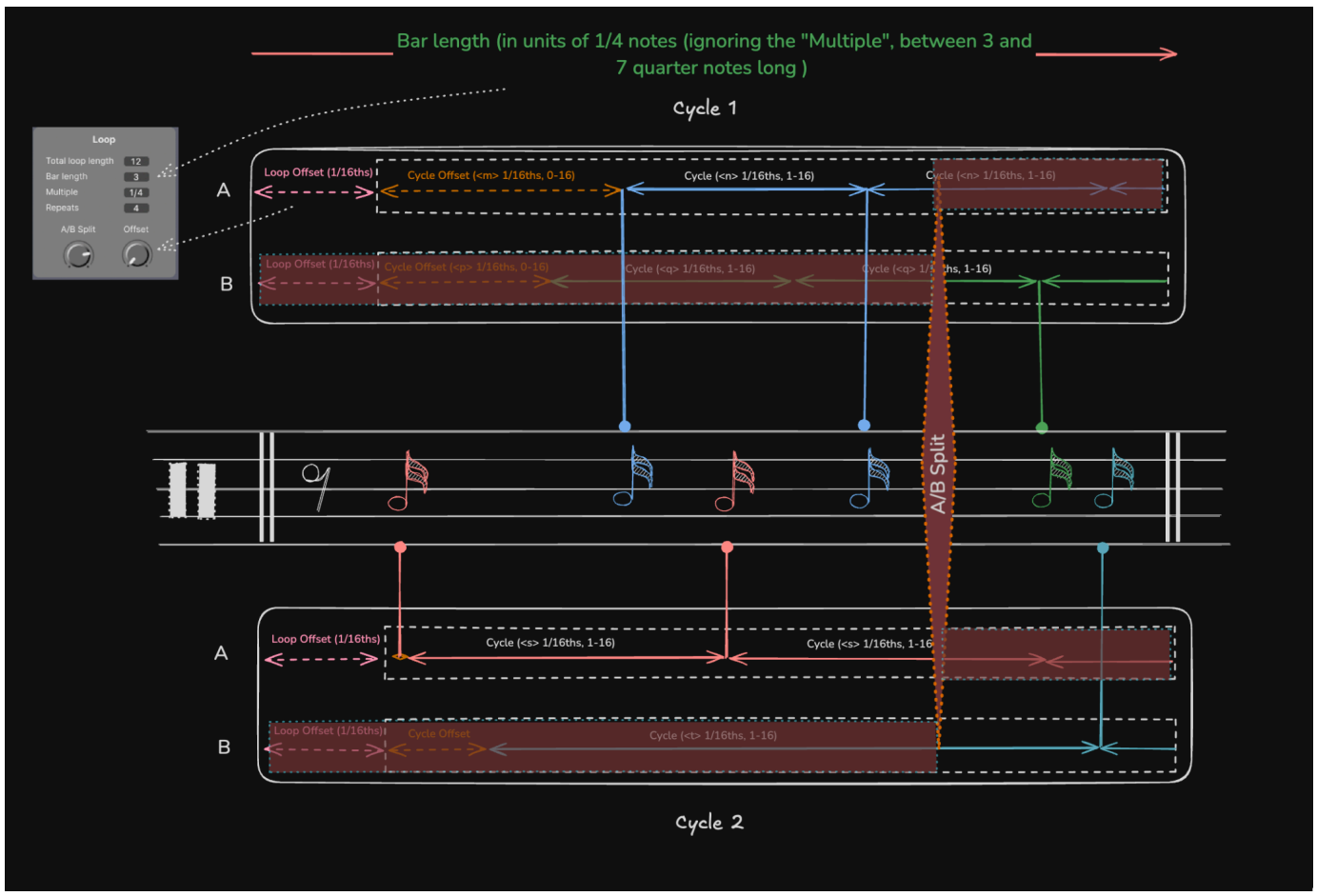


This determines the trigger variation pattern within each cycle part per Cycle loop, using the same algorithms that are used elsewhere in Scapeshift. The Cycle part "Amount" control sets the percentage of variation vs the regular "cycle" trigger stream. The variations themselves will be activated based on the "Variation" section of the Drum engine. This works the same way as in other Scapeshift instrument engines.



As mentioned above, the **Loop** section's "**Offset**" knob applies an additional loop cycle start offset (in 1/16th notes), summed with the offset for each individual loop cycle. This is another great performative control to tweak during a performance.

This illustration might help clarify how Cycles 1 and 2, Parts A and B, and the Cycle and Offset controls all interact with each other to produce drum note trigger events over time.



By manipulating the cycle lengths, split point, and individual modulatable drum parameters, you can design intricate and evolving drum patterns. Although it is not possible to draw a specific sequence of triggers like in a DAW or classic drum sequencer, because these are continuously variable parameters you can seamlessly morph between different drum patterns in the Pattern Sequencer and Wander Mode.

Sound Controls

As mentioned in Chapter 4, the Drum engine is a five voice multitimbral drum synthesizer. It is not sample based, and, as such, you are *not* going to make this sound like a real drummer. But you can make it sound like an amazingly cool & quirky drum machine! Let's take a look.

You'll notice each instrument has very similar controls, while also being slightly different from the others. Before diving into the fun controls like "Thud" and "Whack," though, let's look at the parameters available for each setting.

Value, Cycle, Offset and Amount

By way of example, here is the "Pitch" parameter for the kick drum engine.

The **Value** amount is simply the value of the parameter, pre-modulation. These are all drag controls, so drag it down or up for a lower or higher fundamental frequency of your kick drum. The other three controls are used to **modulate** the set Value.

Cycle controls the length of the modulation cycle, measured in 16th notes.

Offset allows you to delay when the modulation starts–also measured in 16th notes.

Amount controls how much modulation is applied to the parameter.

The settings interacting with each other across all of the drum voices allows for some very expressive rhythmic and timbral changes within a pattern.

Effects Sends

If you look at the far right column of parameters in the Sound part of the drum engine, you'll see the effects sends for each drum voice. And at first glance, they may look all pretty much the same. But there is one important difference to take note of. Here are the effects settings for the Snare (green) and Hi-hat (blue) instruments.

The top FX parameter for the Snare is "> Duck" while for the Hat it's just "Duck". This is because Scapeshift's Ducking effect – a.k.a. The Ducker, described in the "Mix" section above – is hard-wired so that only the Kick and Snare are allowed to control the sidechain (the "Duckers"). Everything else gets potentially ducked by the Kick /Snare (the "Duckees"). So for Kick and Snare, "> Duck" controls how much of the signal will drive the ducking. For the others, "Duck" controls how much it gets ducked. Quack!

The other three controls are pretty much self-explanatory: these are the sends for each instrument to the Reverb, the Delay (actually, the Delays) and the Resonator.

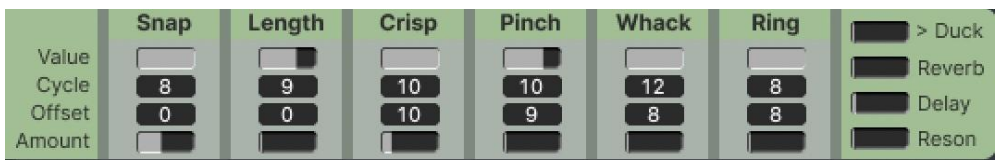
Sound Control Parameters

Before getting started, just a friendly reminder: *all of the listed parameters here can also be modulated!*

Kick

Pitch and **Length** do exactly what you would think they do. **Snap** controls the attack, from dull to snappy. **Thud** provides a low-frequency thump, while **Boom** controls the long decay at the end of the sound. Finally, **Saturate** adds harmonic distortion.

Snare



Snap and **Length** do the same thing that they do in the Kick settings. **Crisp** controls the cutoff frequency of a low-pass filter, while **Pinch** controls the resonance. **Whack** controls the tone of the pitch-based part of the sound, while **Ring** controls the length/release time of it.

Hat

Attack does just what it says, it controls the attack of the hi-hat sound. It has a limited range, so even at its top settings it tends to keep the hat hits in tempo–turning it more from a “tsss” sound to a “shss” sound. **Bright** controls the cutoff frequency of a bandpass filter, while **Pinch** controls its resonance. **Grit** adds bit-crushing, and **Pan** is again hopefully self-explanatory, other than to remind you that modulating it (and everything else here) can be very cool!

Clap

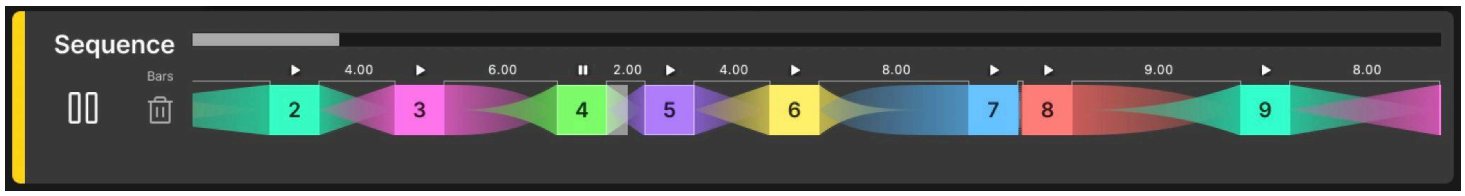
The Clap sound starts with what sounds like four or five different “clappers”. The **Loose** control ranges from them all being in close unison to each other, to them being so spread out that it can produce a castinette-like sound. **Smear** is essentially adjusting the delay time of the sound, ranging from what can sound like tapping or scratching to what can sound like gunshots to–yes–hand claps! **Bright** controls the cutoff frequency of a bandpass filter, while **Pinch** controls its resonance. **Spread** is a stereo depth control (fully down=mono; fully up=stereo), which along with modulated **Pan** controls can lead to some surprising results.

Perc

The Percussion instrument is capable of sounding like anything from congas to a weird contraption made of tuned plastic tubing. The perc engine is a rudimentary Karplus-Strong synthesizer. It features two cross-fed comb filters that are **Spread** by an amount from a central **Pitch**. **Ring** controls the feedback. These are excited by a filtered noise burst: **Speed** controls the filter cutoff, while **Strength** controls the decay of the noise burst. **“Ping”** controls the resonance of the filter.

7. The Pattern Sequencer

Scapeshift features a **Sequencer** that allows you to arrange the patterns you've created and saved to the Pattern Palette.



Not Notes, Patterns!

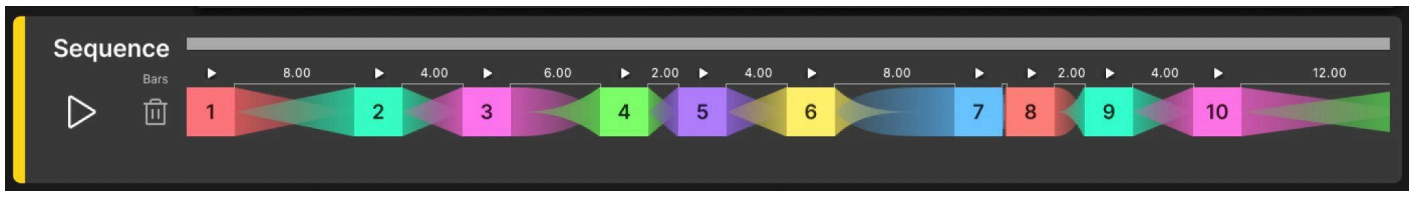
The most important thing to understand about the Pattern Sequencer is this: it is a Sequencer of Patterns. *NOT* of Notes. Which is probably unlike any other sequencer you've ever encountered.

Like any Sequencer, it goes from one event to the next over a timeline. But instead of going from note to note, it is going from pattern to pattern. And if you've made it this far, you know that in spite of all of the Dice and randomness available to you while building your patterns, **Patterns are static:** they play back the same way every time.

The Pattern Sequencer builds on this concept to create something truly amazing, and it's why Scapeshift's "Parametric Design" is crucial. Because everything in Scapeshift is a continuously variable parameter–remember that from the very top of the manual?–the Pattern Sequencer can seamlessly morph from Pattern A to Pattern B, even if the two patterns are wildly different.

At a high level, you build your Song–which is what you're ultimately doing while working in the Pattern Sequencer–by dragging Patterns onto it from the Pattern Palette.

You can drag patterns from the palette onto the sequencer timeline, which works by storing **Pattern States** (snapshots of all parameter settings at a given time) and morphing between them as playback progresses. Here is an example of a pattern sequence:

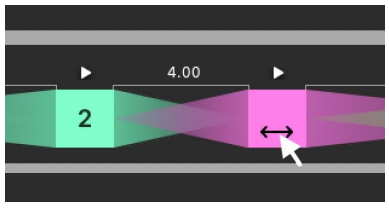


Here we see 10 different Patterns arranged on the timeline, and it's obvious this is only the first part of a longer song, because Pattern 10 is morphing into Pattern 11, which is offscreen at the moment. In fact, this sequence contains 25 Patterns total and runs just shy of nine minutes.

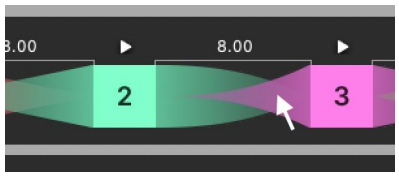
The first thing that should be noted is that the large numbered Pattern Blocks can be visually deceiving, because they have a length of exactly 0:00. If you look for a moment at Patterns 4 and 5 above, you can see that it takes only two bars to morph between them. In spite of the huge boxes, the instant Pattern 3 has finished its 6.00 bar morph into Pattern 4, the playback cursor will start moving from Pattern 4 to 5.

Editing your Sequence

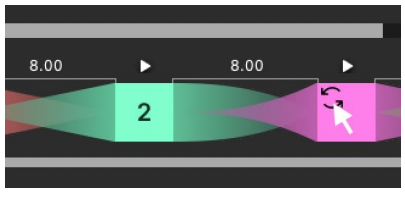
Expanding on the example above, the distance between Pattern 1 and Pattern 2 is 8 Bars, the default length between two patterns when you drag a new one down from the Pattern Palette. Likewise, the difference between Patterns 2 and 3 is 4 bars. To change it, hover over the bottom half of the pattern you want to move until you see the “Move” icon, then drag it left or right to the desired bar length.



You may also notice that in the original example, some of the transition curves between patterns aren’t linear. By grabbing the center of a pattern block in the sequencer, you can change the **curve** of the transition, making the morphing more gradual or abrupt.



You can **overwrite** an existing pattern in the sequence with the currently playing pattern by using the overwrite function.



Amongst other things, this overwrite functionality can be useful for “remixing” a Sequence that you like but want to improve on by tweaking various synth engine parameters (“I wish that Polysynth were a little brighter and the bass a little quieter in this spot!”).

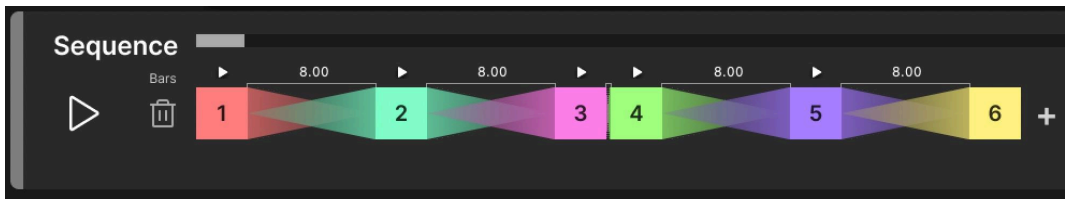
Pattern Placement Strategies

As we have already seen, the sequencer allows for both smooth blending and hard jumps between patterns. And since the entirety of Scapeshift is in play here via Continuously Variable Parametric Design coupled with the morphing abilities of the Pattern Sequencer, just a few strategies with Pattern placement along the timeline can go a long way.

The Transition

As shown in the examples above, you can morph from Pattern A to Pattern B over a set number of bars, with control over the transition curve. This Pattern-morphing technique is probably the one that you will use a majority of the time.

Stationary Jump Cut



Before diving into this example, a note about the colors that the Pattern Sequencer assigns to its patterns when you add them: the colors are fixed and don’t actually mean anything, they’re just to help you visually differentiate between them while you are working with them. They can not be edited or changed.

With that out of the way, here is the layout of the Patterns pictured above:

1 = Pattern A

2 & 3 = Pattern B

4 & 5 = Pattern C

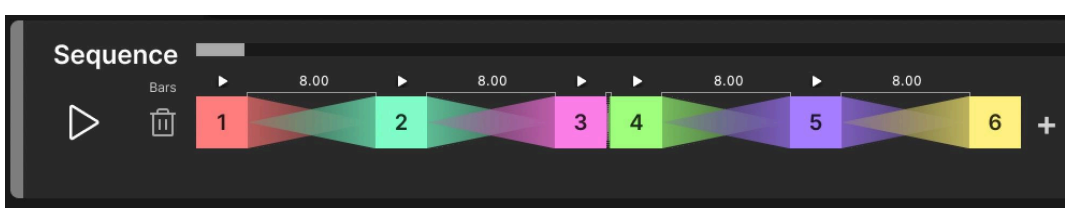
6 = Pattern D

Starting with Pattern A, the Sequence will morph to Pattern B over 8 bars. It will then stay on Pattern B for 8 bars, because it's (not) morphing from Pattern B to Pattern B during that time. At that point it abruptly changes from Pattern B to Pattern C, which continues for 8 bars because it is–again–morphing into itself. Finally, over 8 bars it will morph from Pattern C to Pattern D.

This technique is very useful for controlling and shaping chord change structures, amongst other things, as your sequence/song progresses.

Jump Cut

We can use the same illustration from the section above, but imagine the Pattern layout is different:



1 = Pattern A

2 = Pattern B

3 = Pattern C

4 = Pattern D

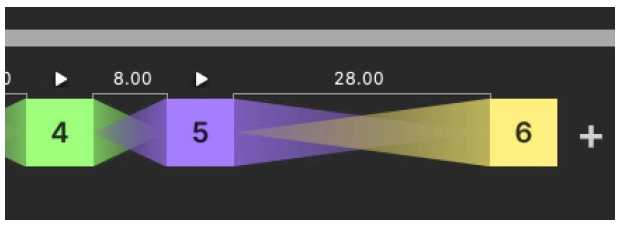
5 = Pattern E

6 = Pattern F

In this case, you will morph from Pattern A to Pattern B over 8 bars, then from B to C over 8 bars. Having reached the Pattern State of Pattern C at this point, however, you abruptly start Pattern D and start immediately morphing to Pattern E over 8 bars. This is another type of transition that you might find useful that is worth experimenting with.

Fadeout Ending

While the Pattern Sequencer isn't capable of performing a fadeout ending on its own, It can give you a long repeating ending figure that you can fade manually. In this example, 5 and 6 are the same pattern, meaning this static pattern will go on for 28 bars before ending abruptly. So you have 28 bars to manually fade away!

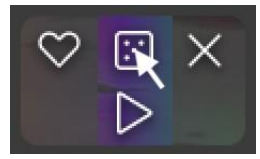


Extended Pattern Palette View

One other tool that is available for building your Pattern Sequences is the extended Pattern Palette view. Access this view by clicking the **“Patterns”** button in the upper left corner of the top part of the screen.

Here, you will see expanded icons for all of the patterns in your pattern pool. The backgrounds of each pattern contains a randomly generated image that can’t be changed. These can help you visually discern one pattern from another, since they don’t have names or labels.

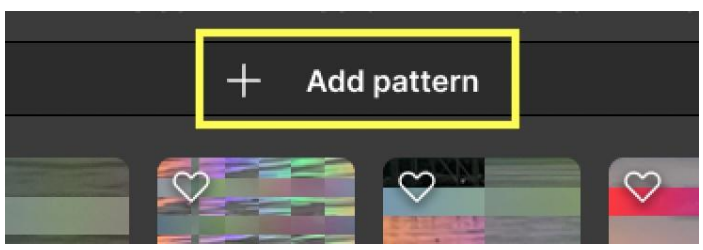
This section isn’t called the “Pattern Editor” because you can’t really edit patterns here, at least not in the ways you would expect. In fact, the only obvious opportunity you have here for change an existing Pattern is with the Dice icon:



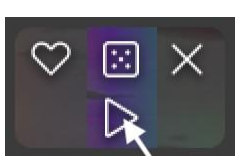
Unlike all the other Dice in Scapeshift, this particular one is not draggable, and it seems to favor adding gentle random changes to the pattern in question rather than generating a wildly different one.

NOTE: Unless it’s already marked as a “Favorite” (see below), clicking this Dice icon will change your pattern immediately within its Pattern Palette slot, and this action is not saved in the Pattern History. Undo is NOT available. Therefore, it is recommended it only be used for patterns you wish to destroy ***or*** *that have been marked as a Favorite.*

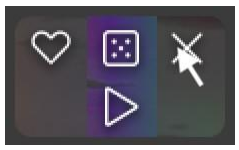
There is a less destructive way of editing your patterns that involves simply playing one of your existing patterns from the palette, changing it in some way, and then saving to the next open slot in your palette with the “Add pattern” button.



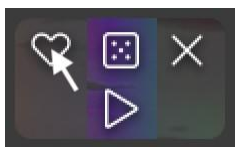
A convenient place to start is with an existing pattern from your palette; you can start playing on with the play button on the pattern’s icon:



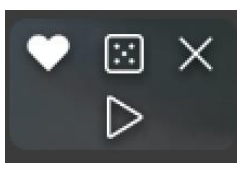
If your Pattern Palette is completely full and none of the 32 pattern slots are open, here is where you can delete unwanted ones to free up slots. Just use the delete icon on the pattern that you want to delete.



Finally, you can mark a Pattern as a favorite with the Heart icon.



When the Favorite icon is lit, the Dice icon next to it behaves differently.



Instead of replacing the contents of the current pattern palette slot with a new randomized pattern, it places the newly randomized pattern in the next available empty slot. This is useful for generating multiple variations on the theme of a single starting pattern.

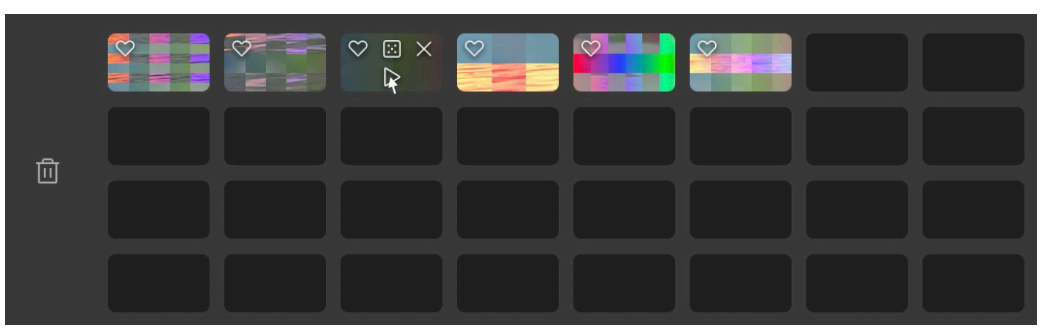
In-Place Pattern Editing

A few paragraphs above, it was stated that the only obvious opportunity you have for changing an existing pattern is with the Dice icon. Luckily, though, there is a *non-obvious* way to change anything you want in a saved pattern. It requires some care and a few extra steps, but it's a powerful little trick to know.

The key to this technique is the way that Scapeshift will always add a new pattern to your Pattern Palette *in the first empty slot it finds.* Normally, the first empty slot will be at the end, after all of your saved patterns. But what if you delete an existing pattern, and now there's an empty slot somewhere in the middle? Aha, this can be exploited as a hidden in-place pattern editing feature.

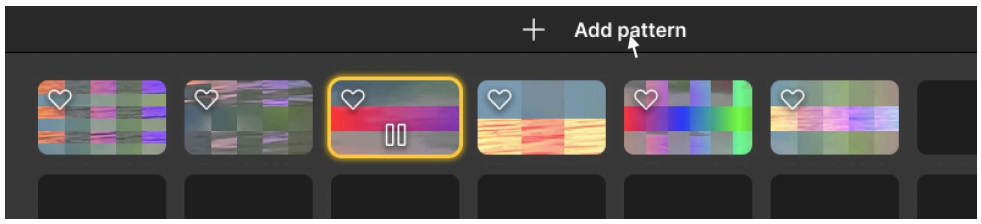
First, a couple of caveats: for editing existing patterns, you'll want to make sure you have Wander Mode (next chapter, we promise!) turned off. Also, after spending some time tweaking and editing, it is very easy to forget which pattern in the pattern palette you're actually editing should focus get lost, which happens frequently. So it may be a good idea to write down which pattern you're working on to avoid having to start over if you lose your place. Something as simple as "2/6" for row 2 pattern 6 will suffice.

Let's take a look at a simple example with a pattern palette that only has six patterns in it, and let's assume we decided pattern 3 would be better with the Bass engine muted. After confirming Wander Mode is off so that randomness doesn't get inadvertently introduced, let's start the process by playing pattern 3 from the extended pattern palette.



At this point you should hear pattern 3 playing and looping. Since the only change we want to make in this case is mute the bass, we do so and are ready to replace the original pattern with our new bassless one. The first step is to delete pattern 3, as outlined in the section above. Now you should have an empty slot 3 in your pattern palette:

Note that slot 3 shows that it is still playing and you should still hear the looping bass-free pattern that you want to save. Click on the “+ Add pattern” button above your patterns to fill the empty slot 3 with the current pattern.



That’s all there is to it! Note that if you already had a sequence built that contained pattern 3 prior to your edit, the sequence will still contain the old pattern 3, *not* the new one. Which brings us to our final revelation about the Pattern Sequencer…

A Truly Hidden Feature

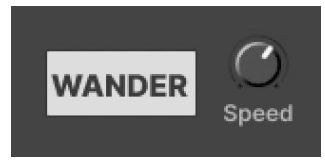
While the pattern palette is typically the main tool used for building pattern sequences, it doesn’t actually have to be used at all. There is no requirement that a pattern in the Sequencer must be saved in the pattern palette. In fact, the patterns in the Sequence are completely independent from the pattern palette, so you can kind of use it (the Sequencer) as an extended pattern bank if you want. You can save up to 64 patterns in the sequence.

While Scapeshift’s sibling product PTNSHIFT is largely beyond the scope of this Guide, the following is worth mentioning here: When you export a your patterns from Scapeshift to PTNSHIFT, all the patterns in the Pattern Palette *and* the Sequence are imported into PTNSHIFT’s note mode. So if you saved 32 patterns in the pattern palette and also built a sequence containing 64 *different* patterns, you can import all 96 of them into PTNSHIFT in one operation.

Same Thing Every Time

It’s worth repeating: The Pattern Sequencer will do the same thing every time you hit play, since it is operating from saved, static patterns. Taking care while building your Pattern Palette and subsequent Pattern Sequence is key to taking advantage of Scapeshift’s morphing capabilities and inherent musicality. In fact, there’s really only one other way to have Scapeshift morph around various patterns and sounds: Wander Mode.

8. Wander Mode



Wander Mode allows Scapeshift to generate continuous variations on your current pattern, which can of course include patterns played from either from the pattern palette or the sequence. When you activate Wander Mode, Scapeshift will alter your current pattern by subtly altering it over time based on the existing patterns that have been saved in your pattern palette.

You can control the **speed** at which these variations occur.

Tim Exile has explained the processing at work for Wander Mode as follows:

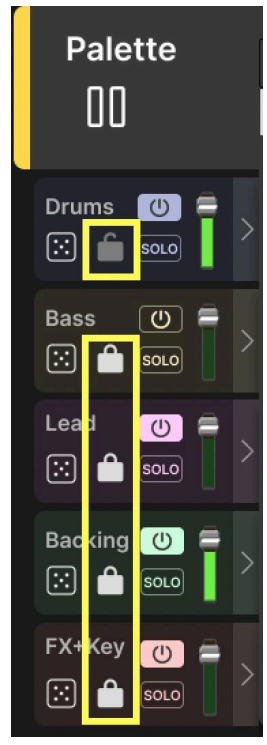
1. Scapeshift will randomly pick a pattern from your pattern palette as the target pattern.
2. It will also randomly pick 10% of the total parameters for morphing; the other 90% will remain static at current pattern settings.
3. Over the period of time set by Speed, the chosen parameters are morphed from the current pattern to the target pattern.
4. The process then repeats, with a new target pattern and new parameters selected for morphing.

While in Wander Mode, you can still manipulate macro knobs and add new patterns to the palette. This feature is excellent for discovering unexpected musical ideas and creating evolving soundscapes.

Not only that, but depending on the patterns that you have curated and saved in your pattern palette, Wander Mode can also be used for generating music in a hands-off manner that may be suitable for improvising along with for live performance. Or just for playing for eight hours solid as background music while gardening, etc.

While working on this documentation, auto-correction suggested changing the spelling of “Wander Mode” to “Wonder Mode”. It definitely has a valid point…

Locks



Locks can be used to “lock” an instrument engine and prevent Wander Mode from changing anything in that engine. Here is an example of where everything is locked except the Drum engine, so in this case Wander Mode will *only* affect the Drums and nothing else.

While on the topic of Locks, the locks also work the same way for things like Theme Card dice and the big “Random Everything” dice found between the History Arrows. Even though they are mentioned almost fleetingly, locks are a useful tool for controlling where randomness does and does not occur within Scapeshift.

Wander vs Dice

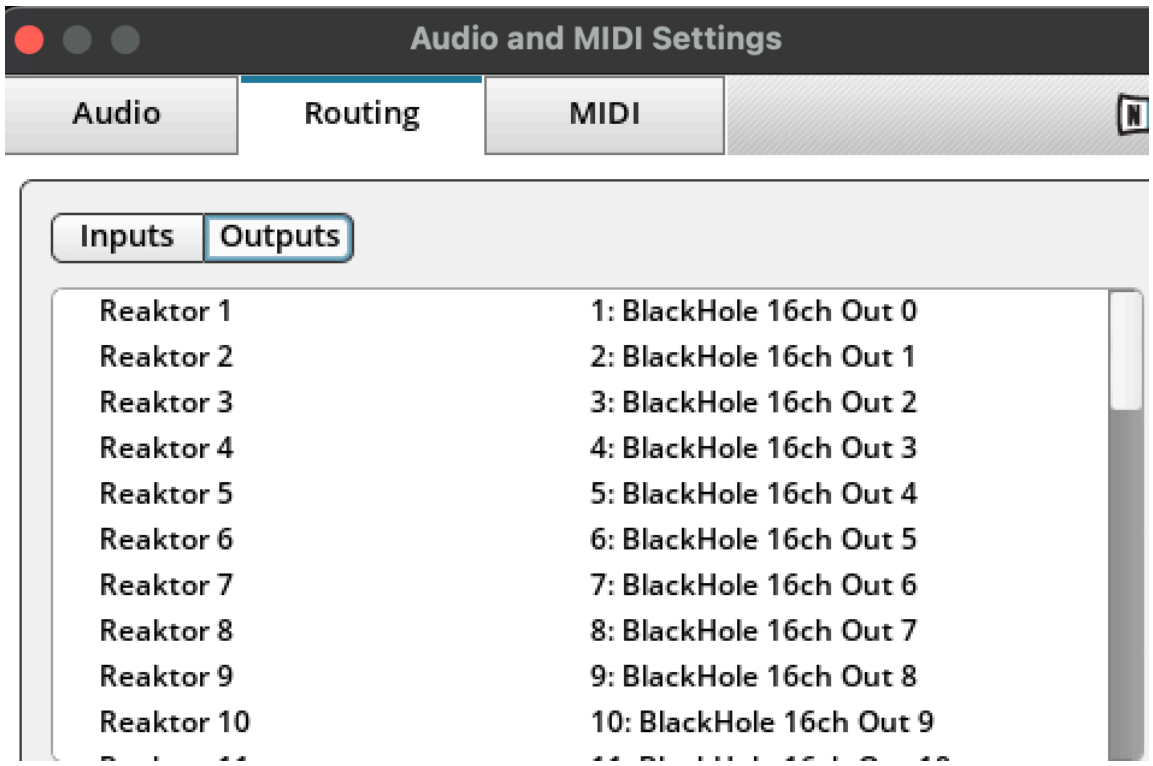
Wander Mode combined with locking is a powerful way to generate new patterns–through mutation over time based on the current contents of your palette pool rather than through the pure randomness of the Dice icons. Let your current pattern wander over time, and when you hear something interesting evolve, you can save it to the pattern palette (or directly to your Sequence) for further use.

9. Standalone Mode

When using Reaktor in **Standalone mode**, Scapeshift can be used to generate MIDI and audio independently of a DAW. This is particularly useful for triggering external hardware synthesizers and drum machines and for routing their audio back into Reaktor for processing. The audio and MIDI settings within Reaktor Standalone allow for detailed configuration of your sound card and MIDI interfaces.

Please refer to the Reaktor User Manual for more information.

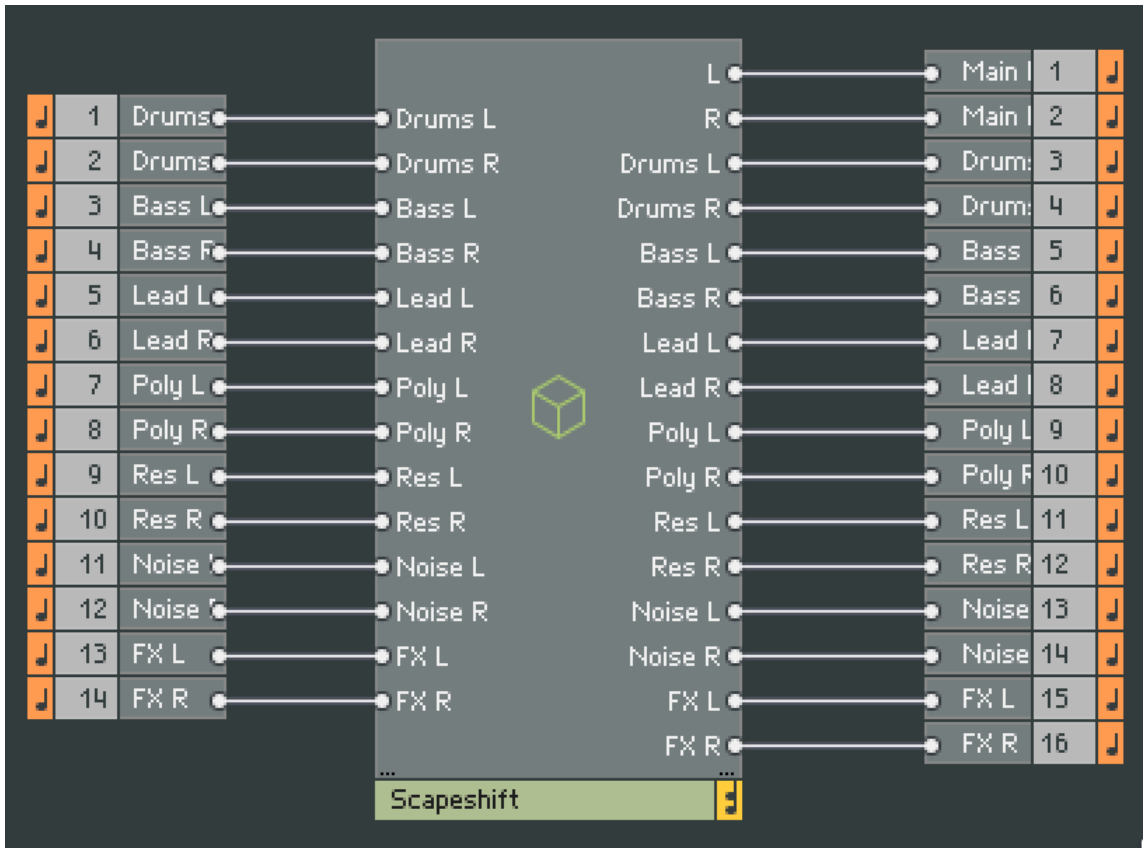
You can use MIDI loopback drivers (as described in Chapter 5 above) to route MIDI out of standalone Scapeshift / Reaktor to other applications in your computer.



You can also use audio loopback (e.g. Blackhole on Mac) to route multi-channel Scapeshift audio output to other audio applications on your computer, as follows:

- Main L & R on outputs 1 & 2
- Drum engine L & R on outputs 3 & 4
- Bass engine L & R on outputs 5 & 6
- Lead engine L & R on outputs 7 & 8
- Polysynth engine L & R on outputs 9 & 10
- Resonator L & R on outputs 11 & 12
- Noise L & R on outputs 13 & 14
- FX (Delay & Reverb, before end of chain effects) on outputs 15 & 16

The external audio input and output port mapping can be seen in the very top of the Scapershift ensemble structure (inputs 1-14 on the left, outputs 1-16 on the right):


1 Drums
2 Drums
3 Bass L
4 Bass R
5 Lead L
6 Lead R
7 Poly L
8 Poly R
9 Res L
10 Res R
11 Noise L
12 Noise R
13 FX L
14 FX R
Drums L
Drums R
Bass L
Bass R
Lead L
Lead R
Poly L
Poly R
Res L
Res R
Noise L
Noise R
FX L
FX R
Scapeshift
L
R
Drums L
Drums R
Bass L
Bass R
Lead L
Lead R
Poly L
Poly R
Res L
Res R
Noise L
Noise R
FX L
FX R
Main L 1
Main L 2
Drum: 3
Drum: 4
Bass 5
Bass 6
Lead I 7
Lead I 8
Poly L 9
Poly F 10
Res L 11
Res R 12
Noise 13
Noise 14
FX L 15
FX R 16

10. Saving your work

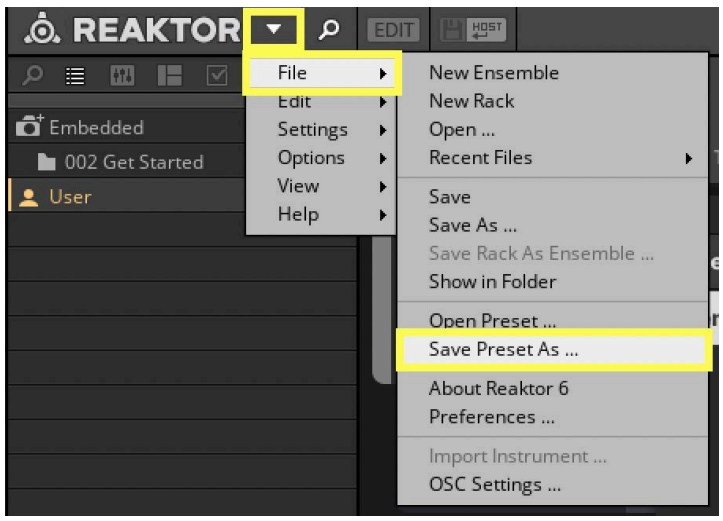
Saving work in Reaktor isn't as intuitive as it is with many other plugins. However, after consulting the Reaktor manual, it's also not difficult. According to the official documentation from the Native Instruments website, there are three ways to save your work:

Save within your DAW Project

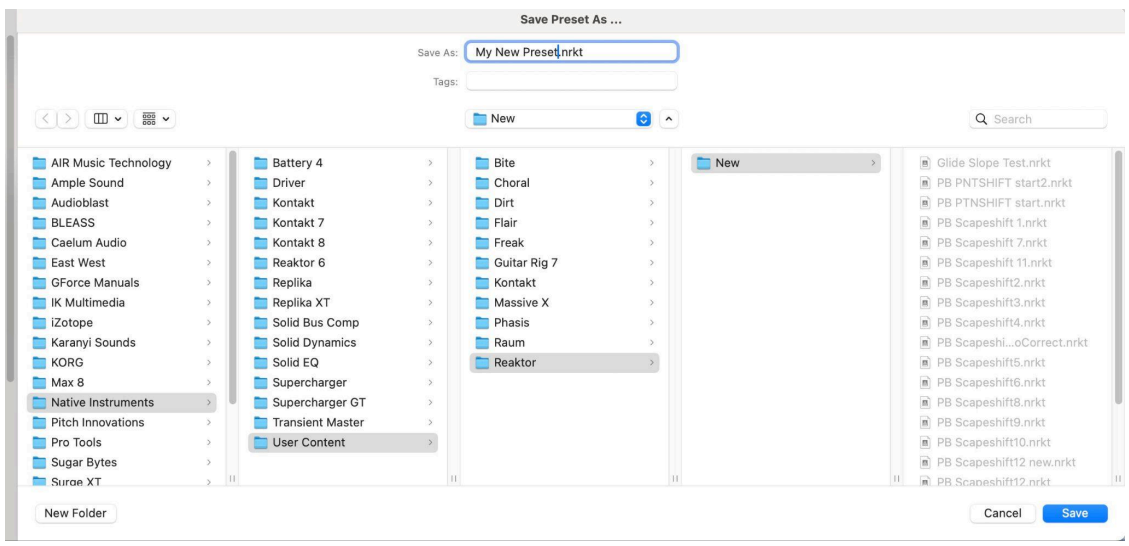
> When used within a DAW, all Reaktor edits and settings get saved within the DAW's Project file. Recall Reaktor's edits and settings simply by reloading the Project file in your DAW.

Save as a User Preset File

> The current set of Scapeshift patterns (in the Pattern Palette and in the Pattern Sequencer) are storable as Reaktor Presets. Scapeshift comes with a selection of built-in Reaktor Presets (each focused around one of the "cards") which may be useful as a starting point. So you can save and recall different sets of Patterns and Pattern Sequencer timelines.
>
> In the Reaktor Menu, go to File > Save Preset As…

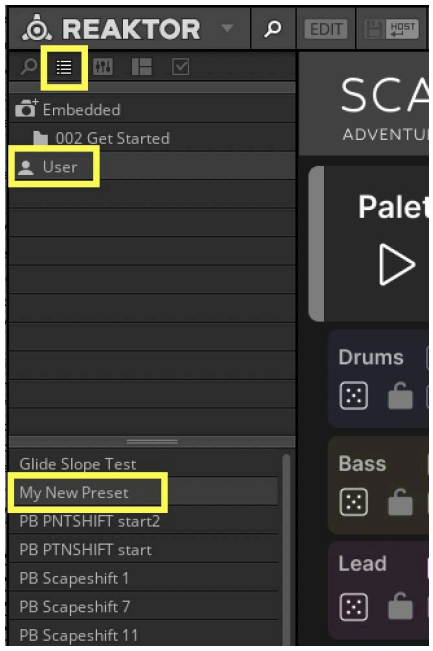


Enter the filename you wish to use in the Save As dialog box.



Native Instruments recommends storing your your .nrkt files in the Users > **Your User Name** > Documents > Native Instruments > User Content > Reaktor file structure, which should be the default and which is shown here.

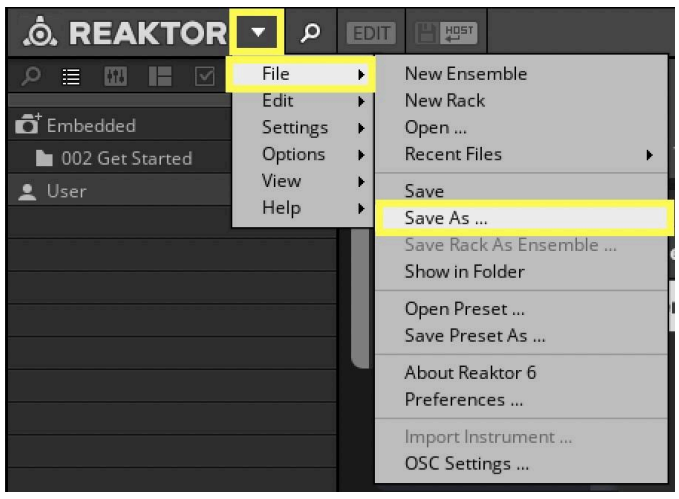
You can then recall your Preset file from the User area in Reactor's Preset Browser.



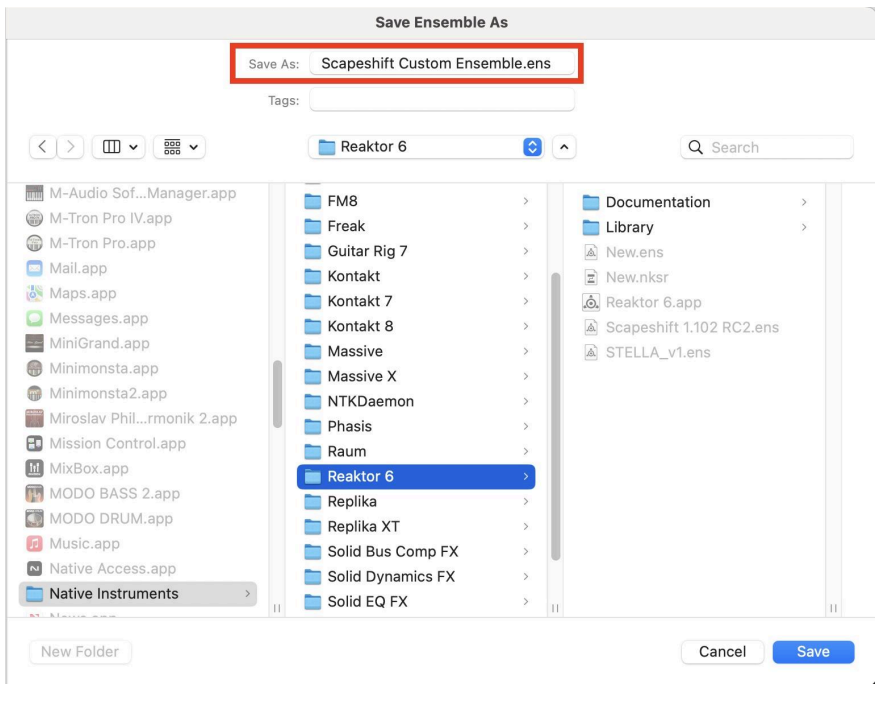
Save as a User Ensemble File

Finally, you can save a new copy of the entire Scapeshift Ensemble that contains your work.

In the Reaktor Menu, go to File > Save As…



Change the filename, unless you intend to overwrite the original Scapeshift ensemble file (.ens).



Now you can start a fresh instance of Reaktor at any time (Standalone or in DAW) and load your custom Ensemble file to resume where you left off in your work.

11. Mastering Your Sound with the Shapeshift Mastering Chain

The Shapeshift mastering chain is designed to provide a cohesive and dynamic final polish to your sonic creations. It is comprised of a series of carefully curated processing stages that work together to enhance loudness, clarity, and overall sonic impact. This chapter will guide you through each stage of the mastering chain.

Dynamic Buses and Dynamic Summing

The initial stage of the Shapeshift mastering chain utilizes a system of **five dynamic buses**: Drums, Bass, Lead, Backing, and Effects.

- Each of these five buses is further broken down into four frequency bands.
- Within each of these bands, a **dynamic summing** process occurs. This system employs **real-time peak detection** to measure the levels within each band.
- The loudest signal within a specific band at any given time effectively **sidechains** the other signals in that same band. If a signal is lower in level than the loudest signal in that band, its level is attenuated, with the amount of attenuation being proportional to how much lower it is.
- This multi-channel, multiband dynamic processing can be likened to a sophisticated form of **multiband sidechaining** where the loudest element in each frequency range dynamically shapes the presence of other elements within that range.
- For instance, if the kick drum has a high velocity and the bass is playing a low velocity note in a shared low-frequency band, the bass may be significantly attenuated by the louder kick drum.
- This process is crucial for maintaining a **tight and controlled low-end** and ensuring that individual elements within the mix have space to breathe without clashing in terms of loudness.

Multiband Compressor (OTT-Style)

Following the dynamic summing stage, the summed output of each frequency band then feeds into a **multiband compressor**. This compressor is designed to apply an **OTT (Over The Top)**-style of mastering, providing further control over the dynamics across the frequency spectrum.

In essence, after the initial stage of dynamic summing, where the loudest element in each frequency band dynamically attenuates quieter elements within the same band, the multiband compressor then takes the summed output of each of these bands and applies a further layer of dynamic processing in an OTT manner. This stage contributes significantly to the final loudness and sonic character of the mastered track within the Shapeshift system.

Tape Emulator (Wow and Flutter)

The next stage introduces a touch of analog character through a **tape emulator**. This effect specifically models the **wow and flutter** characteristics of tape machines, adding subtle pitch and timing variations that can impart a vintage feel. While thought of as a fairly basic tape emulation effect, it is still designed to sound subjectively "right".

Tape Saturation

Following the wow and flutter effect, the signal passes through a **tape saturation** stage. This is a custom-built effect aimed at emulating the **magnetic hysteresis** of analog tape. Tape saturation introduces non-linear harmonic distortion, which can add warmth, thickness, and perceived loudness to the audio signal.

Master Compressor (Long Range)

Further down the mastering chain is a **long-range master compressor**. This compressor is characterized by its **slow attack and very slow release times**. Its purpose is to provide gentle, overall level management, smoothing out broader dynamic fluctuations in the track without overly impacting transients.

Master Compressor (Short Range)

The final stage of the core mastering chain features a **short-range master compressor**. In contrast to the previous stage, this compressor utilizes a **fast attack time**. Its role is to catch any remaining **transient peaks**, ensuring that the overall output level is consistent while avoiding unwanted clipping.

Pre-Mastering Ducking Circuit (Optional)

Before the main dynamic summing stage, Shapeshift also incorporates an optional **ducking circuit**.

- This circuit allows you to send the **kick and snare drums** to a dedicated **peak detector**.
- The output of this peak detector can then be used to **negatively modulate** the levels of the other audio channels.
- This provides a more traditional form of **sidechain compression** that occurs before the dynamic bus processing, allowing for rhythmic pumping effects driven by specific percussive elements.

By understanding the function of each stage in the Shapeshift mastering chain, you can effectively sculpt the final sound of your music, achieving a polished and impactful result. Remember that the specific parameters and intensity of each effect can be adjusted to suit the unique characteristics of your track.

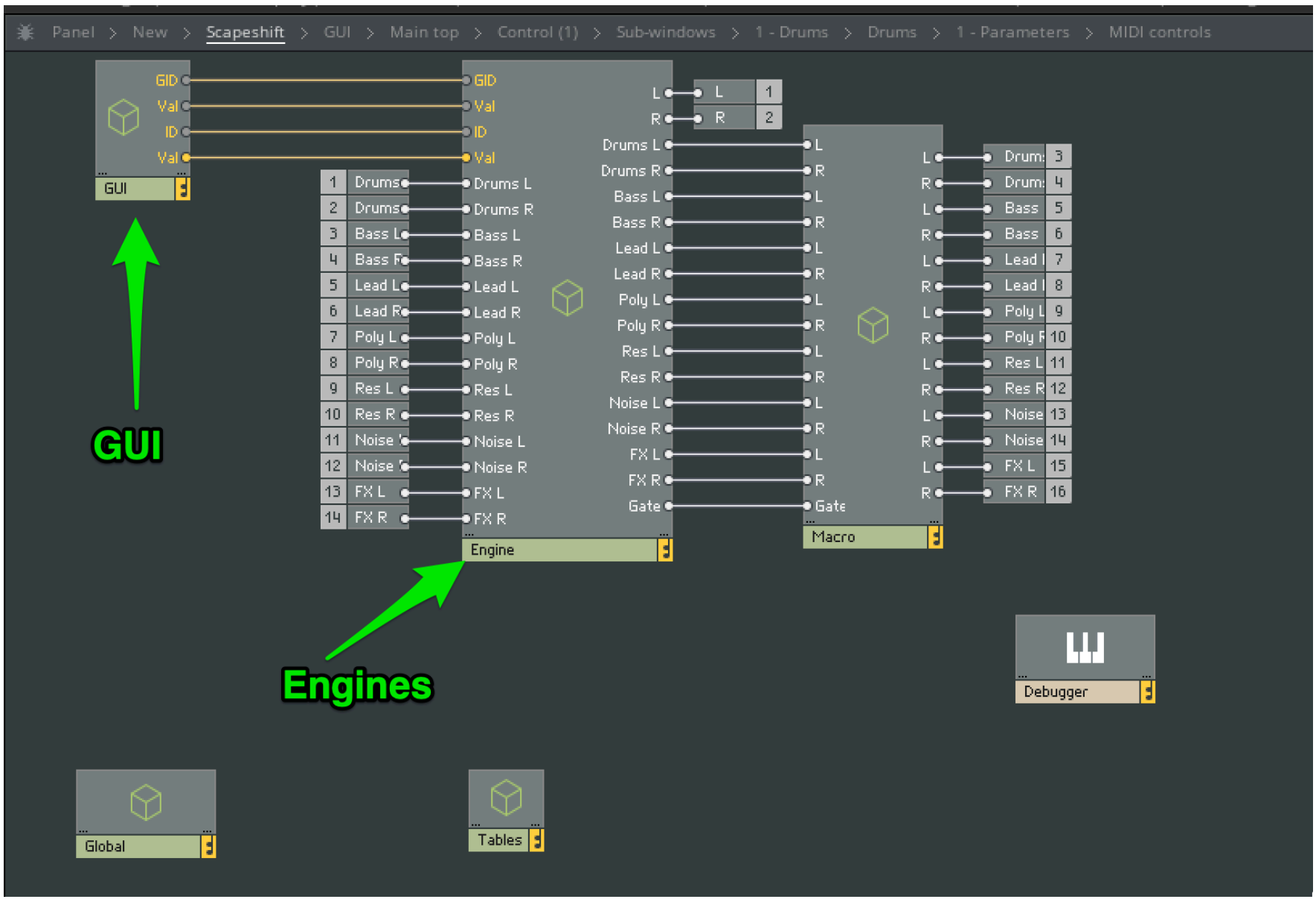
12. Hacking Scapeshift

This section is for those adventurous souls who (a) maybe know a little about how to edit and modify and hack around with a Reaktor ensemble and (b) feel the need to personalise and customise the already awesome sound design potential in the Scapeshift engines with a little extra secret sauce of your own.

We can use this section to share findings and learnings of what is going on inside the Scapeshift monster.

It is strongly suggested you do any hacking experiments with separate copies of the Scapeshift ensemble .ens file, so you can always go back to the official version.

Overall Structure



The Scapeshift ensemble adheres to the well-founded software principle of “separation of concerns”, where all the audio processing is contained in a top-level “Engine” macro, and all the user interface is implemented in a separate top-level “GUI” macro. For any Scapeshift hacking modifications, this means:

1. Any changes to the audio / synth engines must be done somewhere inside that top-level “Engine” macro.
2. Any changes to the Scapeshift user interface must be done in the top-level “GUI” macro.
3. If you add new UI controls (faders etc), you need to somehow transmit their changes into the “Engine” structure.
4. If you want to build new UI feedback (e,.g,. adding level meters to the drum page), you need to somehow transmit the events from the “Engine” structure back into the “GUI” macro.

Tim Exile has recommended on Patreon that no one touches the existing parameter system (it uses 1000s of identified internal id - identified parameters) as it is pretty much maxed out. Essentially this means it is likely not possible–even if we *could* figure out how to do it technically–to add additional engine parameters that are part of the "pattern" save / recall / mutate system. However, we can repurpose existing pattern parameters to do different things, or additional things.

The alternative is to use existing Reaktor mechanisms, used in some of the hacks below (IC Sends, Sends & Receive modules, and just adding more hard-wired connections within the structure) to wire up new audio and UI controls and structures. But these will not work with the Scapeshift pattern system.

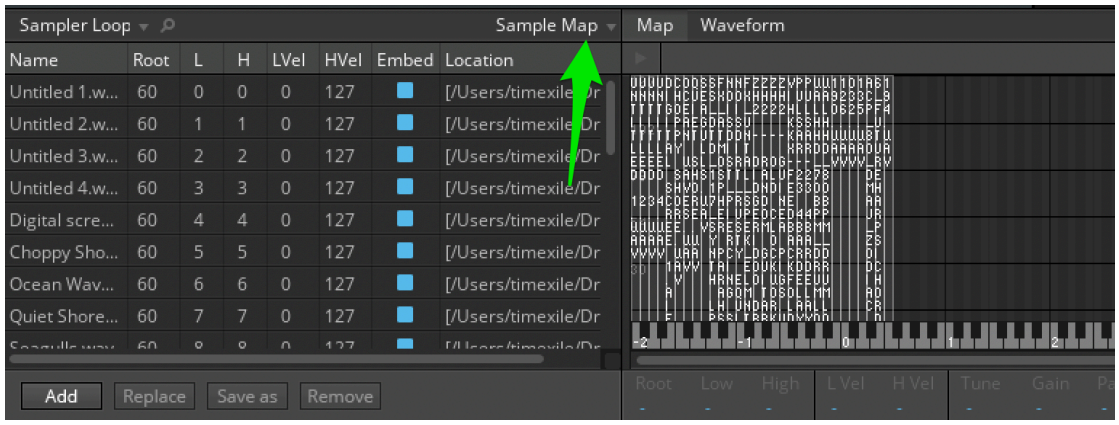
Noise Engine Samples

The Scapeshift Noise Engine uses a Reaktor **Sampler Loop** module at its core, preloaded with a curated set of samples from Tim Exile. But you can replace these with your own soundscapes, field recordings, drum loops, whatever, if you follow the steps below.

1. In your hackable copy of the Scapeshift ensemble, turn on the Edit mode:

2. Click the Reaktor "Show Sample Map Editor" button:

   The Sample Map editor window will appear in a pane at the bottom of the Reaktor UI:

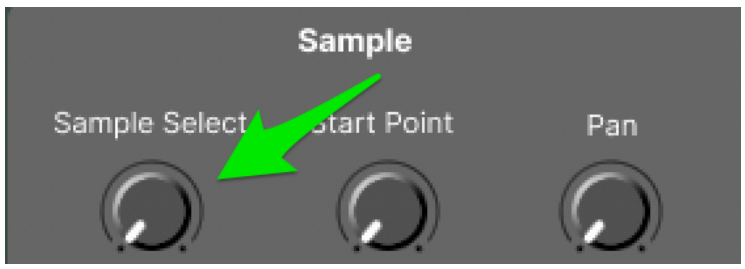


3. There is only one sampler module in the Scapeshift ensemble, so the sample map is displayed immediately. Please refer to the Reaktor user manual for a full description of how to use it. You can load in existing sample maps using the "Sample Map" menu.

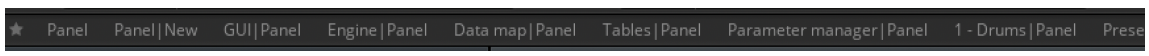
4. For Scapeshift, when you add or replace your own samples (using the Add or Replace buttons), make sure to set each sample's "Root" note to 60 (C4) - the Noise engine assumes this. The L and H values should be the same for each sample, and must be contiguous from 0 to <num samples - 1> in the sample map.

5. OK, now for the fun. The Noise engine's Sample Select parameter is actually a control from 0 to 1. When you move the knob (or when it automatically varies), the sample to play is selected by mapping the value 0 - 1 to the number of samples in the sample set. But we have to tell the

Scapeshift ensemble how many samples there are in the sample map for its use. The default is 8 samples. So if you just curate your own set of 8 samples, there is nothing else you need to do. However, if you want a wider palette to play with, this requires us to edit a single Constant module value in Scapeshift's Noise engine structure.



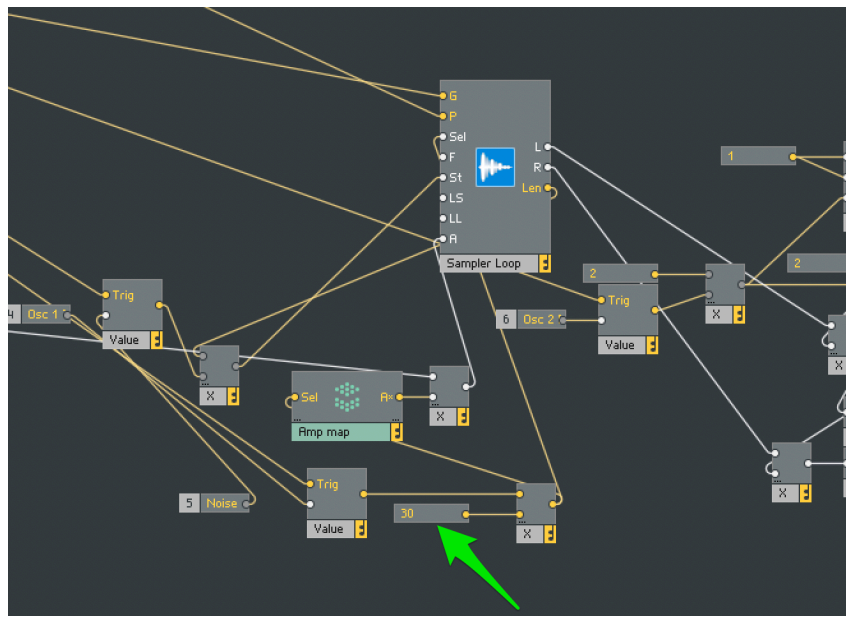
6. If a panel did not open to show the underlying ensemble structure when clicking the **Edit** button in step 1, you may need to right-click on the Scapeshift UI and select the **Show Structure in Other Pane** menu option. Also, select the Reaktor menu **View** -> **Show Bookmark Bar** … as Scapeshift helpfully comes with some useful structure bookmarks which helps to navigate around its extremely complex graph. Click the "Scapeshift" bookmark (Panel -> New -> Scapeshift).



7. Then, by locating and double-clicking on these macros in the structures in turn, you can now navigate to the Noise Engine macro itself:
Engine -> **Noise** (it's hiding in the bottom left-hand corner) -> **Noise engine** (near top right-hand corner). Pause to marvel at the ingenuity and beauty that Tim managed to create as you dive deeper into the structure.

8. It's a good idea at this point to add your own bookmark to the Noise engine macro, so you can then always get straight back to this in the future if you want to modify the size of your sample set again (click the Star button on the bookmark bar whilst you are looking at the Noise engine macro structure):

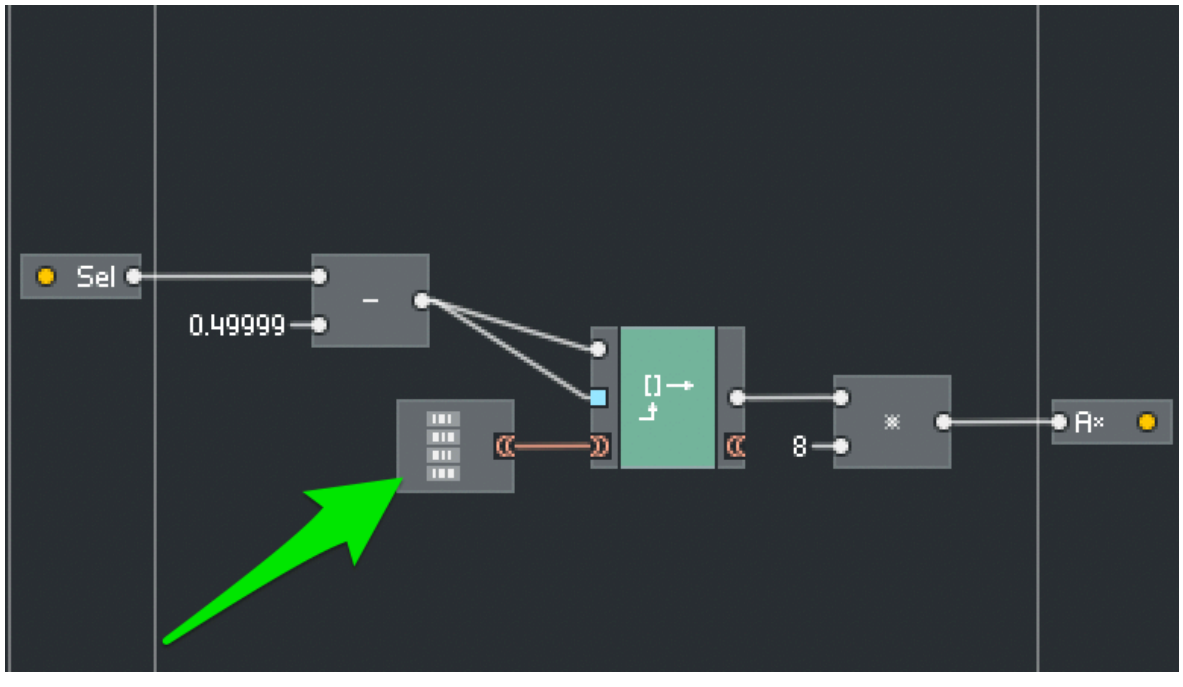
9. Almost there! Now find the Sample Loop module in the Noise engine macro structure, then find the Constant module pointed to in the screenshot. That's the Constant that tells Spaceshift how many usable samples there are in the Sampler Loop's sample map. For an OG Scapeshift v1.06 ensemble, this will show **8**. Double click the module, type in the sample size you want to use and hit Enter to make the change. It can be equal or less than the number of samples in your sample map. If it is greater than the number of samples in the map, you will just hear no sound if those non-existent samples are selected by the Sample Select control (which could be an interesting creative effect…).



10. Save your now-modified Scapeshift hacked ensemble file. Congratulations! you can now play and experiment with your own custom samples in your Scapeshift productions.

WARNING! As mentioned above, by changing this "sample map size" constant, you are changing the mapping of the Sample Select 0 - 1 value range to the samples in the map. So this will likely alter which actual samples are selected and played for any existing Scapeshift patterns. Which again can be a cool creative tool in itself.

11. For bonus points, if you want, you can figure out how to edit the embedded core table in the nearby **Amp map** macro which sets the relative amplitude at which each sample in the map is played by the Sampler Loop module. The default seems to be "1" (original amplitude) so mostly this is not needed.

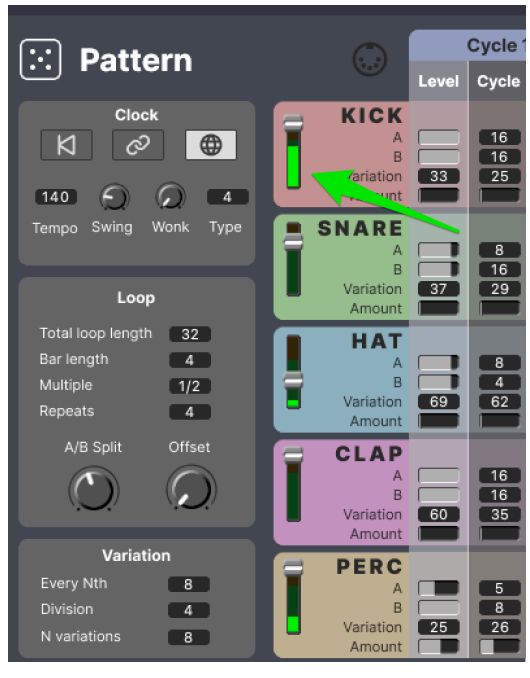


12. Now, you can turn off Edit mode, close any additional structure panes and the Sample Map pane in the Reaktor UI, navigate over to the Scapeshift UI Noise engine tab, and experiment with a wonderful new source of sounds to add to your Scapeshift universe.

Adding Drum Voice Level Meters

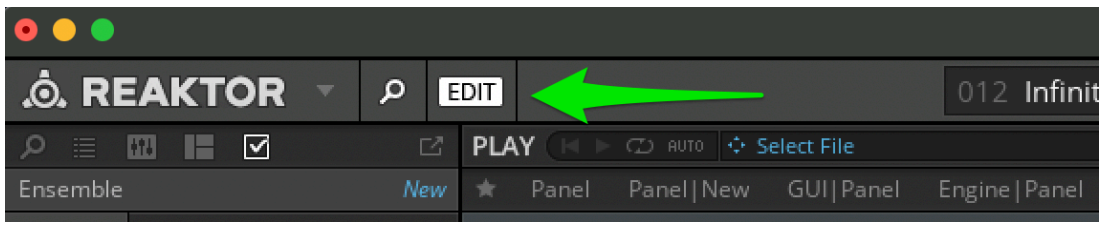
In Scapeshift v1.06 (current version at time of writing this), the individual drum voices have their own level faders, but it is not currently possible to see the levels coming out of each Voice separately. This hack adds level meters overlaid on the existing drum voice faders as shown in the screenshot above. It would really help if you have a basic working knowledge on navigating around a Reaktor structure and simple Reaktor UI panel editing knowledge for this hack, especially trying to follow this written recipe, but fortune favours the brave…

IT IS DIFFICULT TO DESCRIBE SOME OF THESE STEPS - PLEASE LET US KNOW ON DISCORD IF YOU GET STUCK!



In summary, we are going to co-opt the Scapeshift Meter Cache mechanism to send the audio levels from each drum voice (in the “Engine” macro) to new level meters buried in the top-level “GUI” macro (see the “Overall Structure” subsection above).

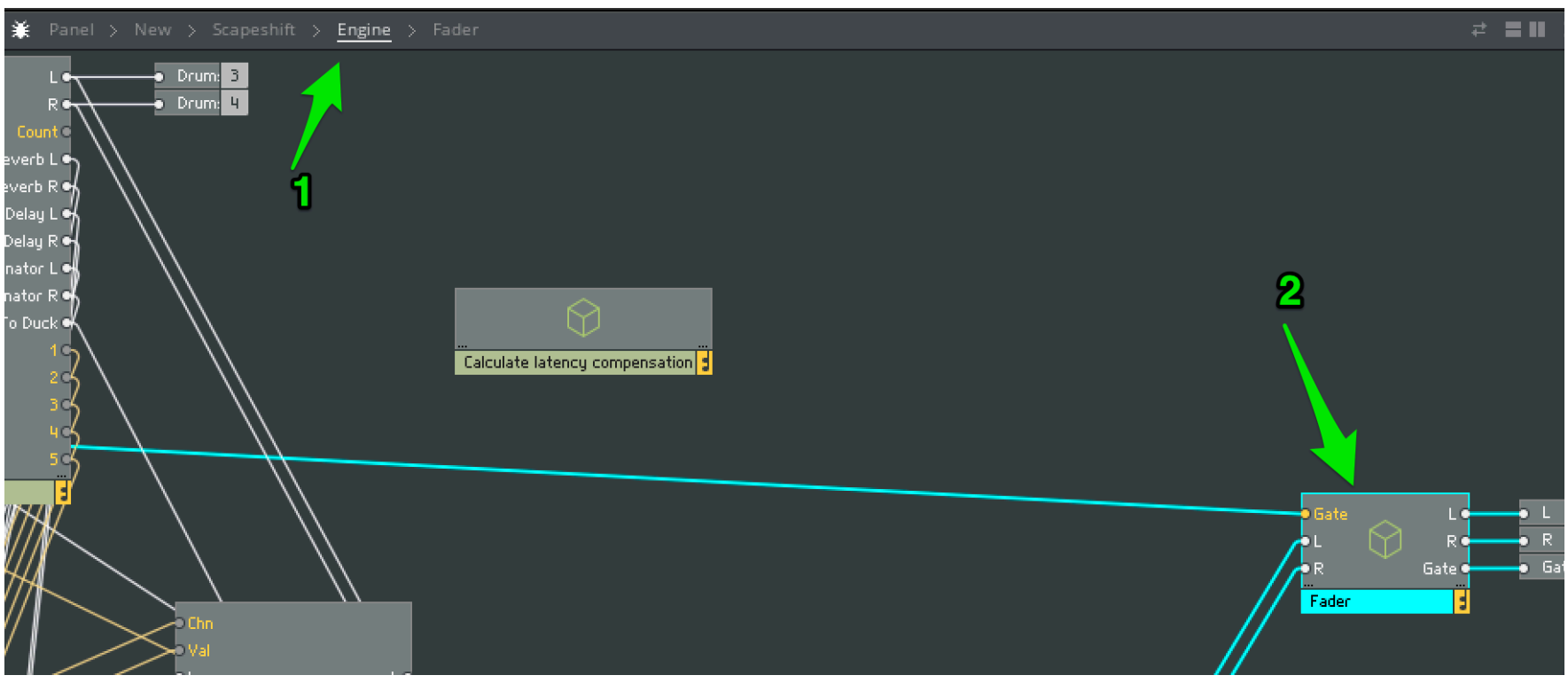
1. In your hackable copy of the Scapeshift ensemble, turn on the Edit mode:



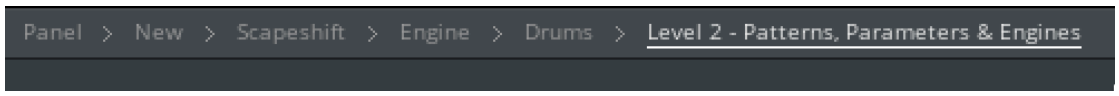
2. Open another Reaktor pane to show the Scapeshift structure in one pane and the Scapeshift UI in the other pane:

3. First, we need to copy an existing Macro in the Scapeshift structure.
   Navigate into the main “**Engine**” macro structure, and find the “**Fader**” macro (top left-hand corner)

   Click into the fader macro, find the “**Write Meter Cache**” macro, select, right-click then “Copy” (to copy it into the Reaktor clipboard). Double-click on the Fader background to return to the Engine macro structure (or click on the “Engine” bookmark again):

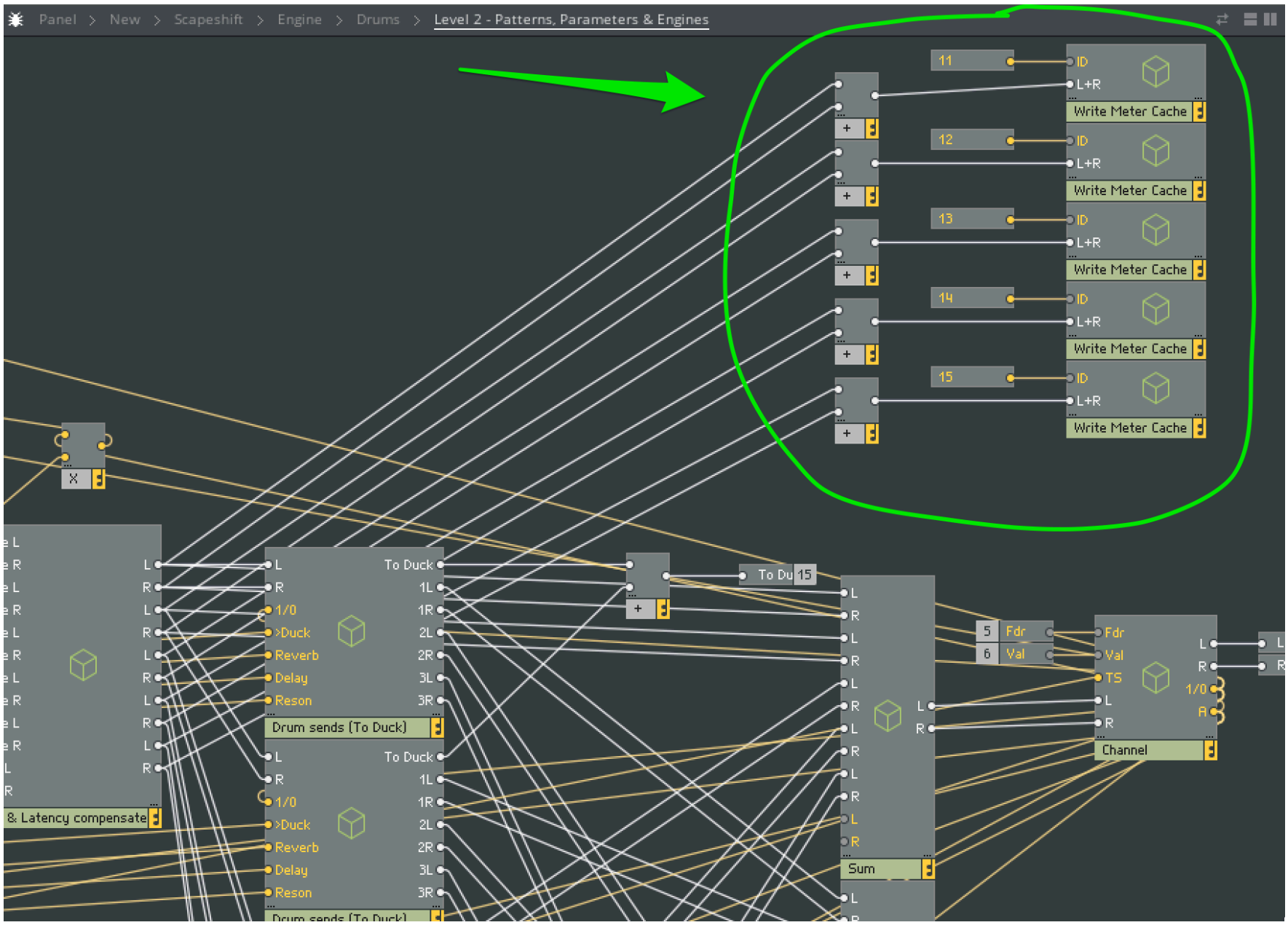


4. Navigate down through the “Engine” -> “**Drums**” macro to its “**Level 2 - Patterns, Parameters & Engines**” macro structure:



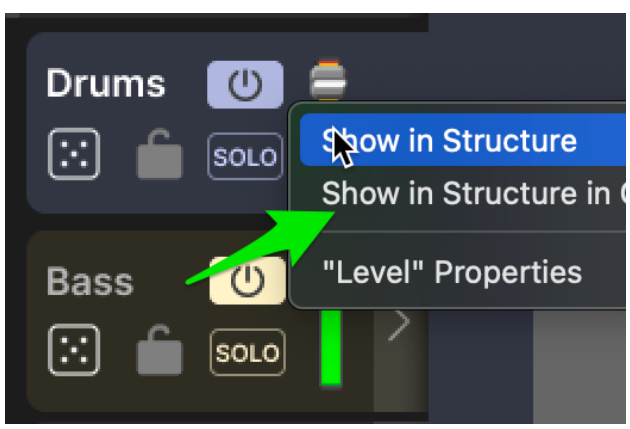
5. Wire up the new structure shown below in the top right-hand corner, above the final “Sum” and “Channel” macros. Paste in the “Write Meter Cache” macro from step 3, then duplicate it again three more times. Wire the L & R outputs from the “Route & Latency compensation” macro to the new Adder modules.

This new structure writes the drum voice audio out signals (combined to a single mono signal) into new entries (11 - 15) of the system-wide "Write Meter Cache" (a shared table of values used as a cache for metered values).

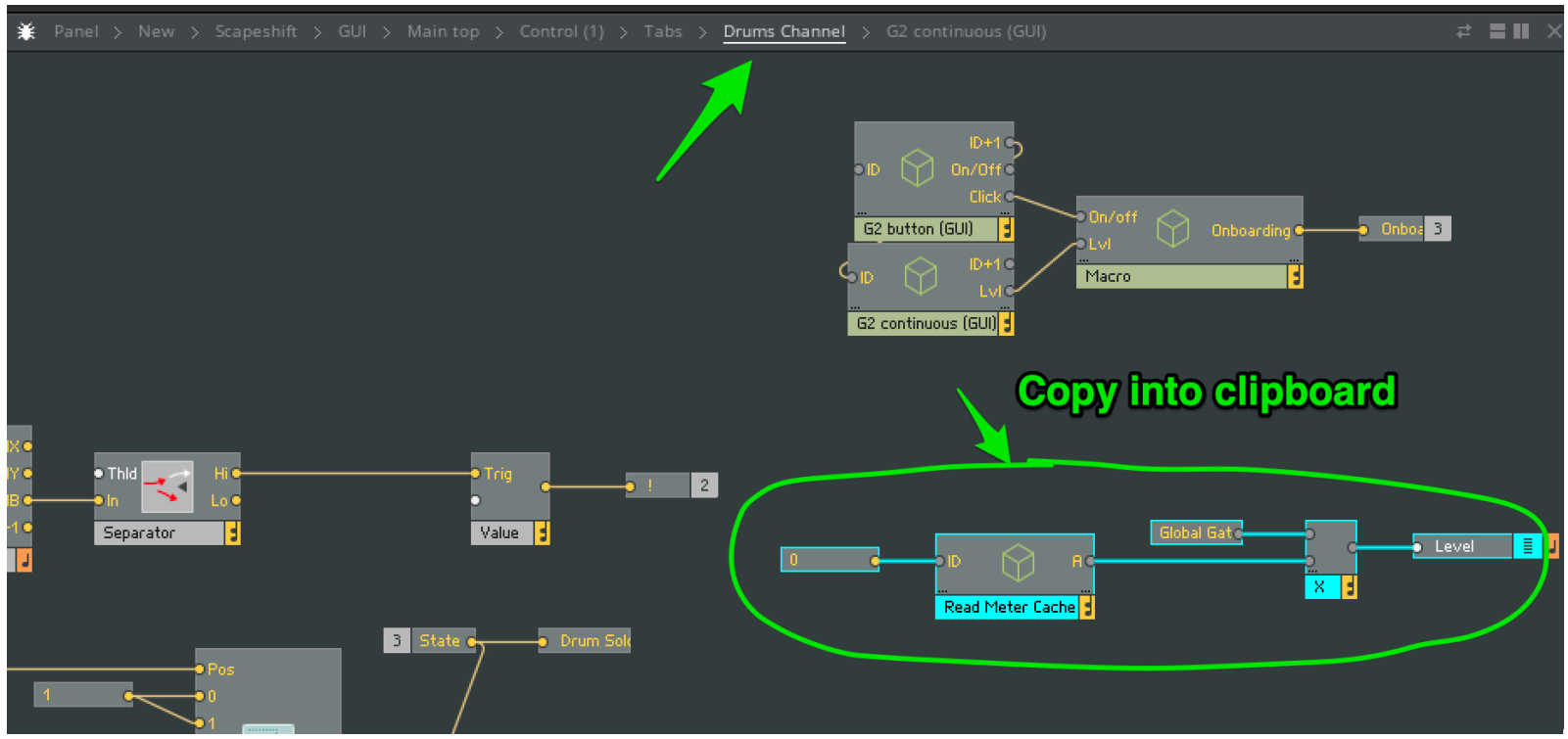


6. Now we need to read these cached meter values out of the cache again and show them on UI level meters. This gets a bit tricky to describe on the printed page (vs showing on a YouTube video), but here we go…. First we need to harvest a copy of another existing macro in the Scapeshift ensemble.

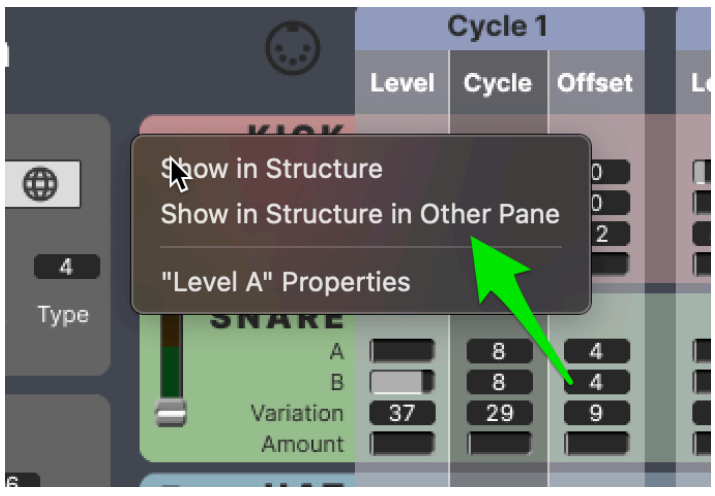
    a. In the Scapeshift UI pane, right-click on the left-hand "Drums" tab level meter, select "Show in Structure in Other Pane".



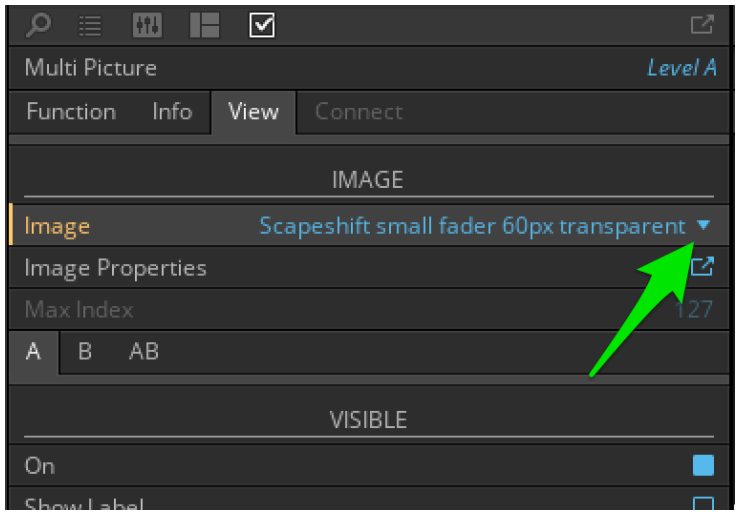
    b. Double-click on the background to go up one level to the "Drums Channel" macro structure. Find the mini-structure with the"**Read Meter Cache**" macro (near top left-hand side), select all of it, right click "Copy".



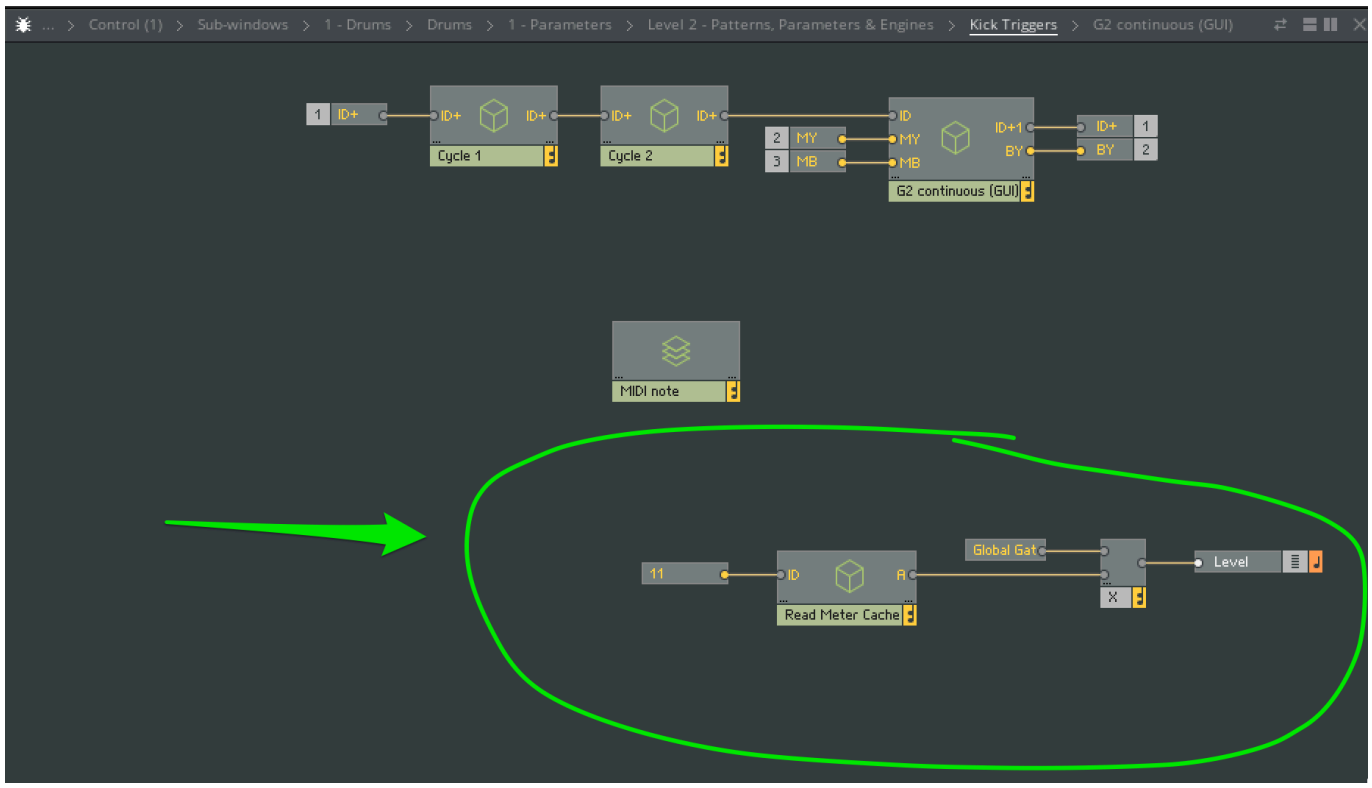
7. In the Scapeshift UI pane, open the Drum engine tab, then right-click on the fader on the Kick voice, select “Show in Structure in Other Pane”.



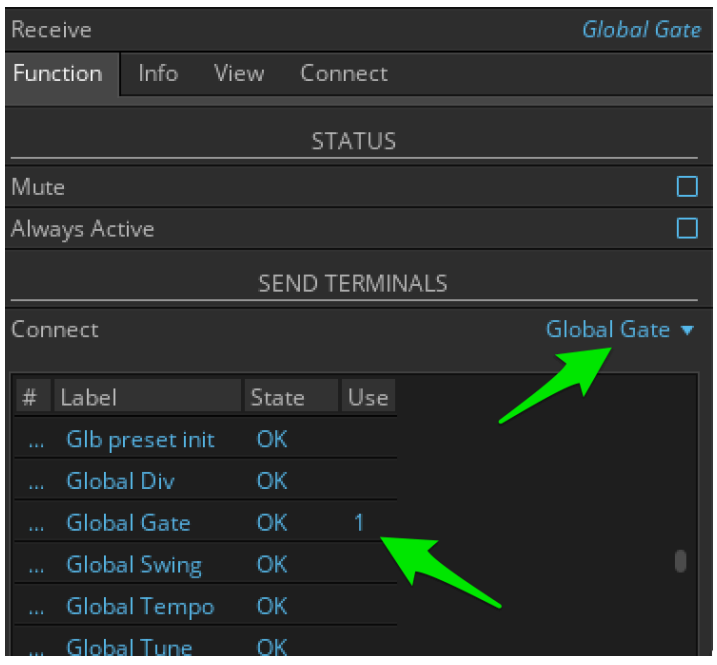
8. This should open the “Kick Triggers” -> “G2 continuous (GUI)” macro structure. Select the “Level A” Multi Picture module, change its Image (in the View properties tab) to “Scapeshift small fader 60px transparent”. This lets us visually place the new level meter underneath the (transparent) fader image. Double-click on the background to go back up one level to the structure of the “Kick Triggers” macro…



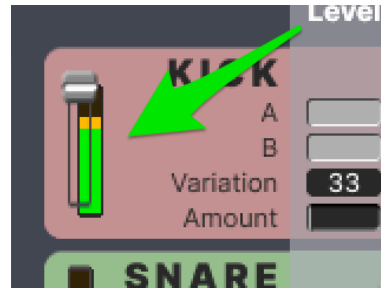
9. Paste the structure from step 6 into the “Kick Triggers” macro as shown below. Change the Constant to **11** (for the kick drum meter), (then use 12 for the snare meter, 13 for the hat, 14 for the clap, 15 for the perc meter).



The “**Global Gate**” is a Reaktor Receive module, make sure it is connected up properly (this will turn off the meter if there is no pattern playing (either from the Pattern Palette or the Sequencer).

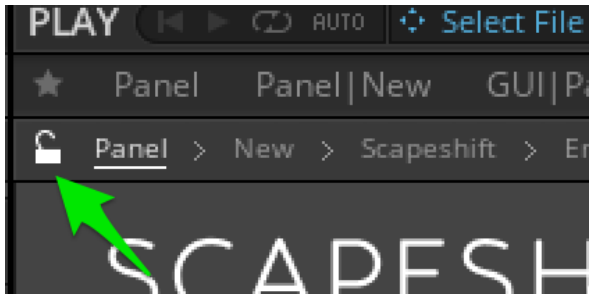


10. At this point the Scapeshift UI might reset to the top-level state, so you will need to open the drum engine UI again (this happens a lot when editing the Scapeshift UI). When you do so, hopefully you will see something looking like the picture to the right:

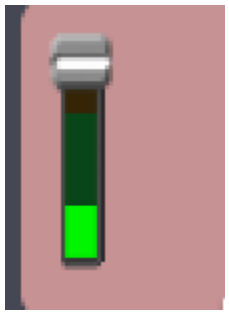


**** SAVE THE ENSEMBLE FILE AT THIS POINT - when moving elements around on the Scapeshift UI on the next steps, it is easy to move the wrong thing like the engine background. Undo might save you, but better to reload your progress up to this point again****

11. Unlock the Lock Panel toggle button and turn OFF the Reaktor menu Settings -> Snap To Grid:



12. Select the new level meter and using keyboard arrow keys, nudge it so it is positioned on top of the fader (but because of the fader transparency, it will look like you are sliding it underneath):



13. Now repeat steps 6 - 12 for the **Snare**, **Hat**, **Clap** and **Perc** drum voice faders & level meters, saving your ensemble file as you go just in case. The “Triggers” macro will be different in each case, but they all have the same structure.

14. Finally, lock the UI panel again, and save the ensemble once more.

**** CONGRATULATIONS … YOU SHOULD NOW HAVE A COPY OF THE SCAPESHIFT ENSEMBLE WITH WORKING DRUM VOICE LEVEL METERS! ****

Adding CC Values in MIDI Mode

This hack will allow you to send MIDI CC values out alongside MIDI Note values from a Scapeshift engine when it is in MIDI Out Mode. The Polysynth is used in this example, but it will work for other engines as well.

Prerequisites: this hack uses two Reaktor macros that were written by Phommed, one of the authors of this guide. Download these files, which are hosted from Tim Exile’s Discord server, for use in the steps below:

- Poly MIDI CC out.mdl

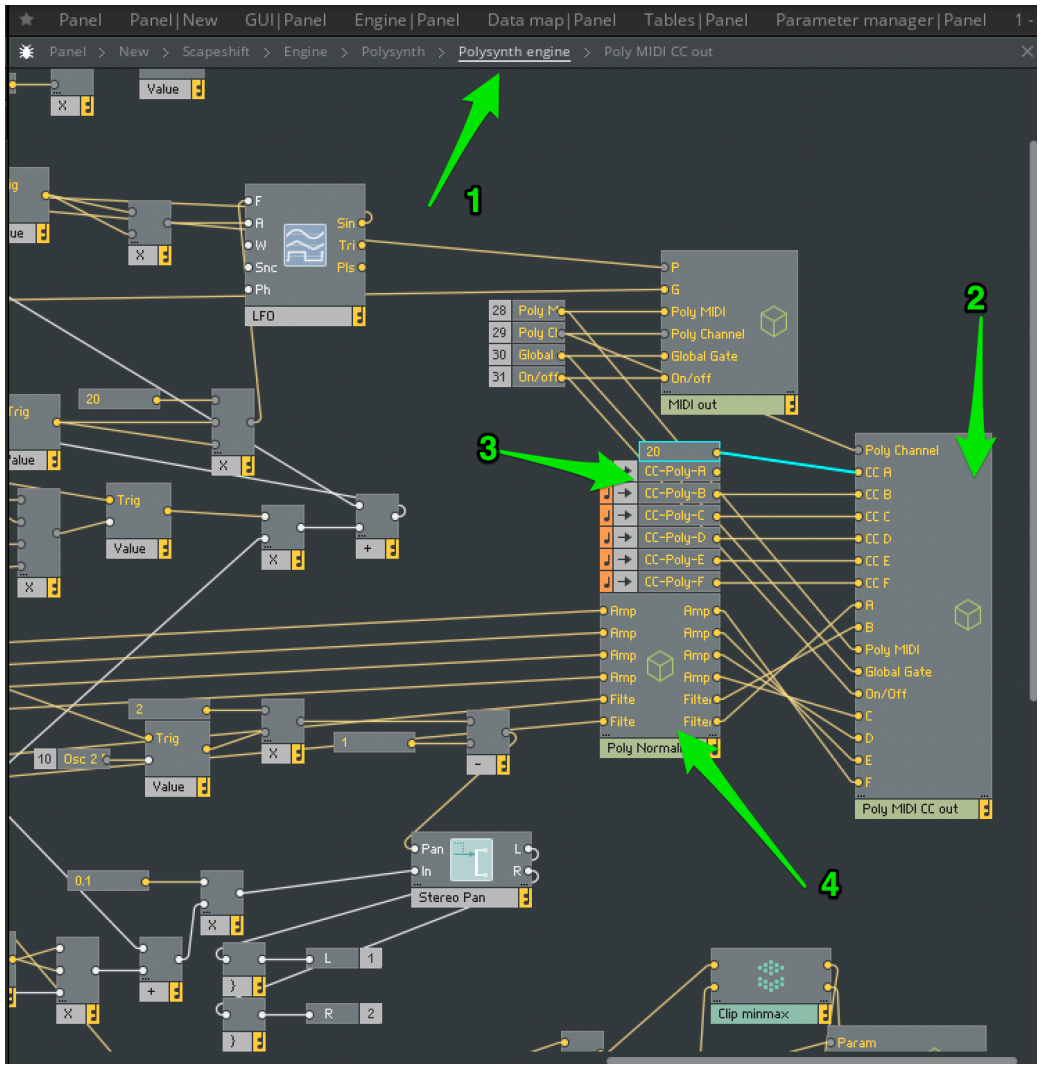
  https://cdn.discordapp.com/attachments/1308494067423903775/1358116850927538306/Poly_MIDI_CC_out.mdl?ex=68027e61&is=68012ce1&hm=1aa4b8f2081aeb7cc75bafa0ff1adde37104bba91e4eccdc63773787c2c84ee4&

- Poly Normalise Values.mdl

  https://cdn.discordapp.com/attachments/1308494067423903775/1358116851275792545/Poly_Normalise_Values.mdl?ex=68027e61&is=68012ce1&hm=111a6c2aae95ec01e2fcad18b9ef568319146e3164f707e2c015db7df64d6604&

  These links are both broken…did the files get hosted elsewhere? –PB

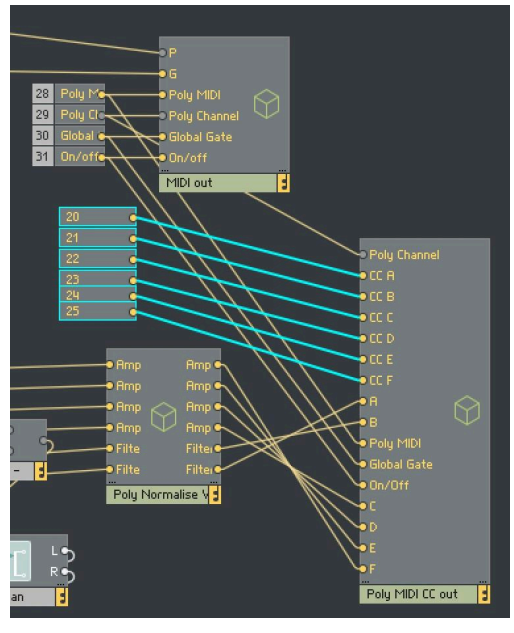
Using your hackable copy of Scapeshift, here are the steps, as annotated in the following screenshot:



1. Within the Scapeshift structure, navigate to the Polysynth engine. Go to the top right-hand corner in the structure, near the "MIDI out" macro.

2. Drag the "Poly MIDI CC out.mdl" Reaktor macro (see link above) into the structure. This macro will convert six engine parameters to separate MIDI CC messages.

3. The next step is to wire up the MIDI CC numbers you want to send. These feed into the six “CC A” through “CC F” inputs to the Poly MIDI CC out macro. In the screenshot above, Phommed was experimenting with some GUI controls that are beyond the scope of this hack, which are going to inputs B-F. However, his example does use a Constant for input A, which is hard-wired to send MIDI CC 20. We will expand on that (contract from that?) by simply using all Constants instead to hard-wire the CC numbers.

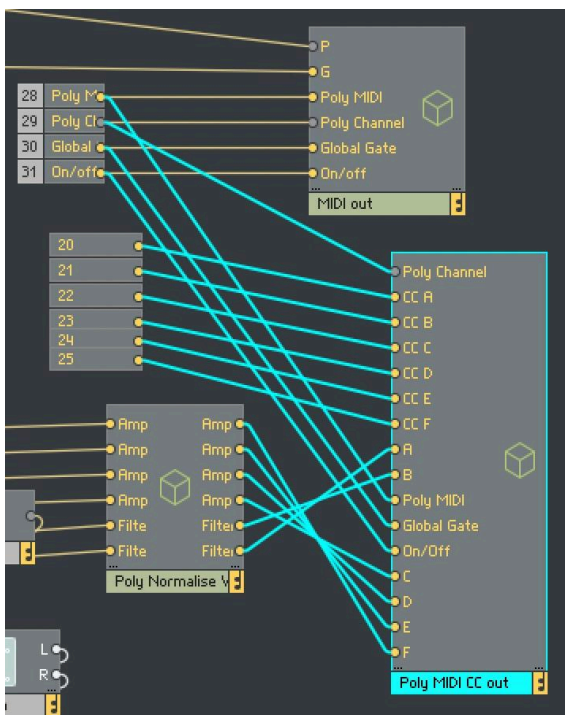
   

   To hard-wire all of the CC numbers that you will be using, pictured is an example of using six constants to send CCs 20 through 25. Add six Constants (right click the background where you want to add them, then select Built-In Module > Math > Constant) and set the values to your desired CC numbers and wire them up now if you wish:

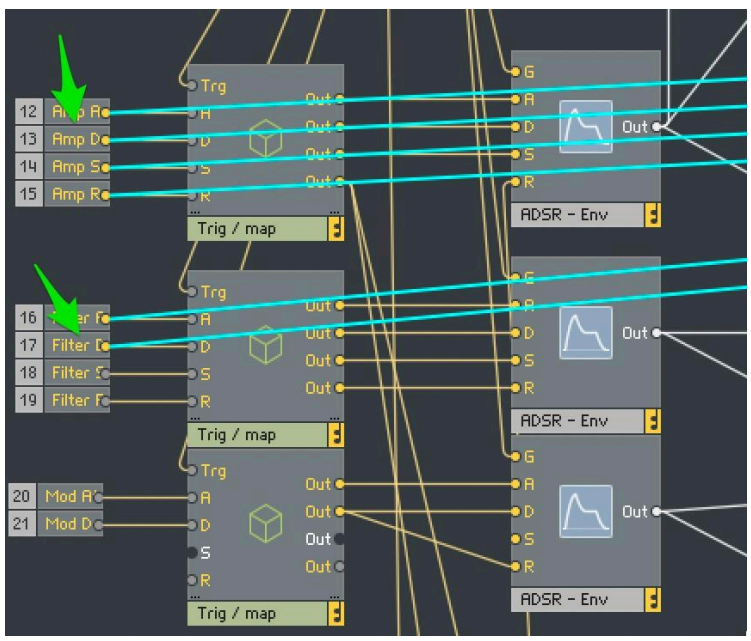
4. The A - F value inputs in the lower portion of Poly MIDI CC Out expect events with values normalised in range 0-1. The “Poly Normalise Values.mdl” macro (link above) was written to help you

do exactly that. Drag your downloaded copy of Poly Normalise Values.mdl into the structure as pictured in the main image above.

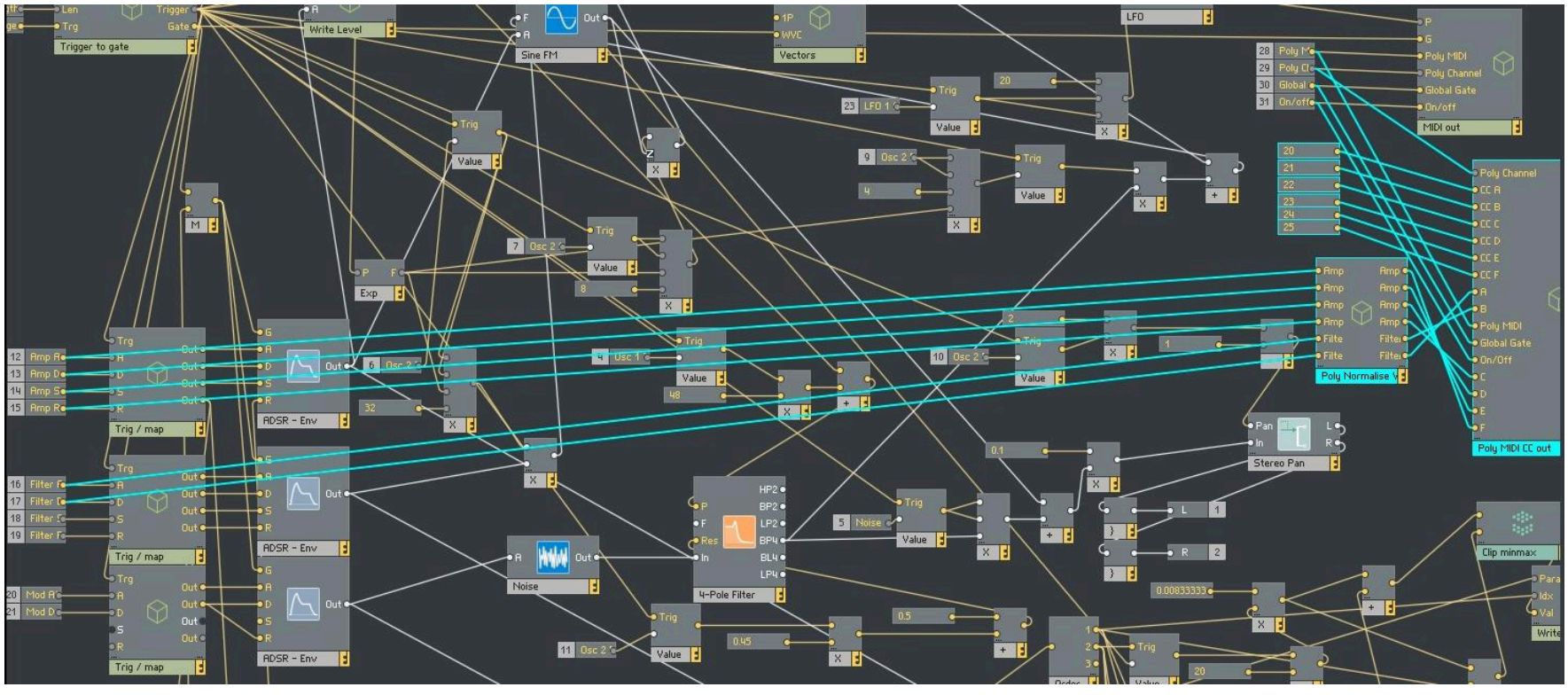
5. Wire up the “Poly MIDI CC out” inputs, as shown with the blue lines below (the “MIDI out” object was moved slightly to clarify the wiring):



6. Finally, wire up the Polysynth engine parameter inputs to your Normalise macro. For this example, we will use the "Amp Atk, Dec, Sus, Rel" (which are the Osc1 envelope controls) and the "Filter Atk, Dec" (which are actually the Osc2 Atk and Rel). You can find them way over in the bottom left-hand area of the structure as pictured here:



These will be connected to the “Amp” and “Filter” inputs of the Poly Normalise Values macro. The completed wiring job covers a lot of real estate, but it will look something like this:



7. Test your hack by turning on the Polysynth engine’s MIDI Mode, then connect a plugin or hardware device to the appropriate MIDI channel–default for the Polysynth is MIDI ch 4–and set the parameters you wish for Scapeshift to control to receive CC 20-25.

Testimonials

Pink Bob

Hello! I'm Pink Bob. I'm one of the writers employed here at Tim Exile Corporate Headquarters, typing from the rather unglamorous 17th floor (You're on a floor? Luxury! ed. note)

No, no, not really…I'm joking of course. I'm just a devoted user of Scapeshift–and its sibling PTNSHIFT–and a volunteer for something that I truly believe in, at a time when the world is providing me not much else that's positive to believe in.

Tim Exile Patreon Adventurers plug

I've really enjoyed working on this manual with Tim and The Introvert and Phommed and Positron Alpha and…my, the list does go on and on!--and all the members of the close-knit family that is (currently, early 2025, ed. note) Tim's Patreon Adventurers Club.

*Tim Exile's Discord server is THE place to be if you're into Scapeshift/Ptnshift and want to sit at the Genius Table for lunch every day like I get to. (I'm the dumb guy at the table, believe me!) Tim **really is** building all of this by himself, with no employees or even a room to call a proper office at this point. So we pitch in from around the world where and how we can.*

Consider joining us to learn, share, hack, teach, and contribute, as well as help Tim chart the future direction of Scapeshift and its related products:

https://www.patreon.com/c/timexile/membership

What do Birds have to do with anything?

Tim's fond of rambles, so I'll continue my own here and then try to eventually bring things back on topic. When I got Scapeshift on New Years Day, 2025, I had a three day learning curve–at about ten hours a day. On the fourth day, I wrote a studio track with Scapeshift that shocked me and changed my whole outlook on how music could–and perhaps even should–be made. That track is called "Birds".

*I share this not to plug my obscure and until this point intentionally anonymous anti-commercial music, but rather as an example–NO, AS A TESTAMENT!–of what the Pattern Sequencer can do to literally change your life. I mean, seriously, just listen to Scapeshift in all its **Generative Glory** here:*

https://pinkbob.bandcamp.com/track/birds

While the "lead drummer" is a UJAM plugin that is independent of Scapeshift, Scapeshift's generated Drum engine starts things off and is key to the overall feel of the rhythm track throughout. In fact, the UJAM drums don't even come in until about 1:54. Crucially, everything was generated by Scapeshift via the Pattern Sequencer except for the UJAM drums and field recordings of birds.

Before getting back to our regularly scheduled programming, I further confess to writing the "Multitrack Recording with Scapeshift in a DAW" and "Pattern Sequencer" sections earlier in this manual. They document exactly how I am using Scapeshift to record fully-realized studio tracks–with a rather tedious but ultimately very rewarding workflow. Incredibly substantial tracks (at least in my mind) have been made using this method.

For me at least, the Pattern Sequencer is the most original and important part of a delightfully original and important Instrument. I mean Workstation. Er…System?

/Pink Bob

The Introvert

Having used more or less esoteric hardware sequencers and DAWs since the late 80s, Scapeshift is the first truly interactive composition tool I tried that I actually “clicked with”. There is something very rewarding about working with an environment that is able to surprise you while still being true to the boundaries of the music you are creating and actually add or embellish to your stuff, much like a fellow musician would.

That Soundscape is not only helping you create variations and transitions from your patterns and rhythms but is also able to morph and modify the actual sounds *themselves makes it so much more powerful.*

While investigating and learning how to use Scapeshift, I created the progressive techno album “The Future we were Promised” where Scapeshift produced a lot of the underlying structure with me playing additional keyboard parts on top and arranging the songs.

If you’re curious to what Scapeshift can help create, It can be found on Bandcamp, here:

https://theintrovertmusic.bandcamp.com/album/the-future-we-were-promised

and on Spotify:

https://open.spotify.com/album/6MKpOoBVdHzT8IHfooFqvD?si=Tej5SQhjQI6kCmmU2HpQsg

/The Introvert

Contents