Nestup Max for Live Device Reference

Foreword

Hello! If you're reading this you've probably downloaded Nestup. Thank you so much for supporting this strange and slightly manic attempt to do something new with musical rhythm. Nestup started as a series of what-if questions. What if Ableton Live fractional time signatures? What if you had a septuplet grid? What if you wrote your own language for rhythm from scratch? The end result is an ongoing experiment that is successful in some ways and goofy in others. So before we dive in, we just want to say thank you for coming along for the ride and being part of this discovery process.

Are you having trouble? Find our **troubleshooting** section below.

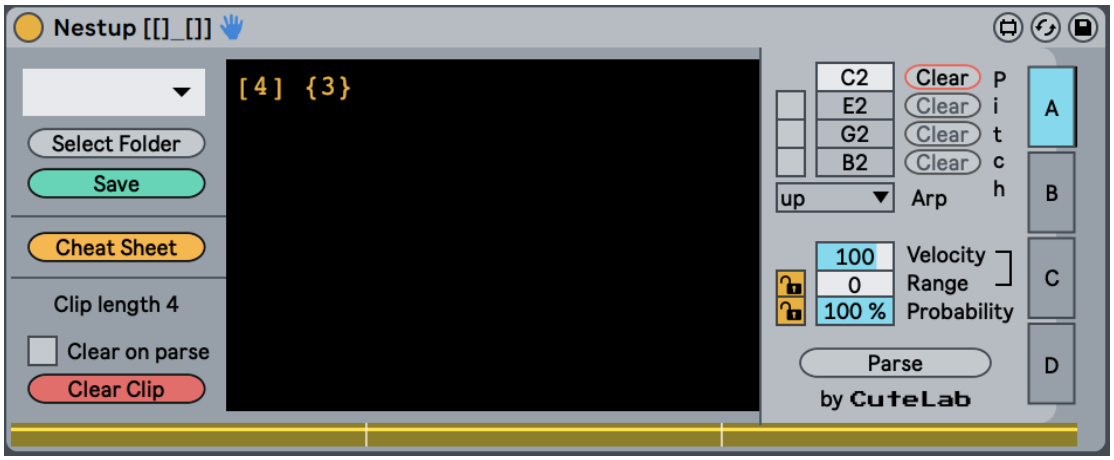
Intro

Nestup is both the name of a language for musical rhythms, as well as the name of this Max for Live device. The goal of the device is to interpret the language and to interface with Live. We'll usually refer to the device as Nestup, and to the language as the Nestup language, or Nestup expressions.

Nestup is a MIDI device, but it doesn't alter or generate a MIDI stream. Instead, it generates a MIDI note pattern and writes it automatically to a MIDI clip. To get started with Nestup, click on the center code box. Then enter a Nestup expression. You might try:

```
[4] {3}
```

Then, press the "Parse" button in the bottom-right of the device, or press Shift + Enter. You'll see a colored pattern appear at the bottom of the device.

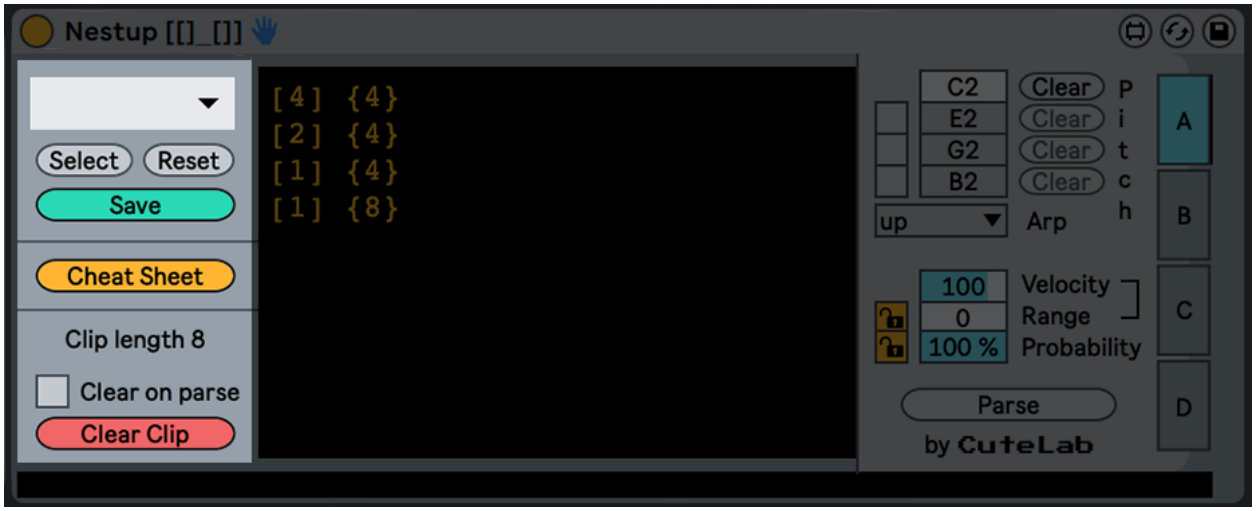


The colored pattern is a preview of what's been written to the first clip slot of the track. If you switch to Session View and look at the first clip slot, you'll see your pattern written to that clip. A new clip will be created if there wasn't one there to begin with, otherwise notes will be added to the existing clip.

See the appendix for a guide to the Nestup language. Or, you can visit the project homepage on GitHub https://github.com/cutelabnyc/nested-tuplets.

Prewritten Patterns

Nestup is divided into three major parts. On the left there are controls for picking from prewritten patterns, as well as options that affect the overall functionality of the device.



At the top of this left part of the device, a menu lets you choose from a number of pre-programmed patterns. This is a great way to see what Nestup can do, and to familiarize yourself with the language. Beneath that, the **Select Folder** and **Save** buttons let you work with your own patterns. **Select Folder** will let you choose a new folder as the source of the pre-programmed pattern menu. This can be any folder on your computer, and the menu will be filled with any text file in that folder. You can press the **Save** button to save the current pattern in the Nestup language editor to your computer, **Reset** will return you to the built-in folder. Nestup files are text documents, but the preferred file extension is .rhy for "rhythm."

Next, the **Cheat Sheet** button will open the Nestup language documentation to the quick reference. This highlights the "best of" the Nestup language and gives you a quick reminder of how to create most kinds of patterns.

Finally, the bottom of this area deals with the overall clip. The **Clip Length** indicator simply tells you how many beats were last generated from the expression in the editor. Beneath that, the Clear controls handle clearing notes from the clip in the first clip slot. **Clear Clip** simply clears all MIDI notes from the clip. The **Clear on Parse** toggle changes what happens when you parse a

Nestup expression. With this control enabled, Nestup will clear all other notes in the clip before adding new ones. Otherwise, it will only clear notes with the same pitch as Nestup, and will leave all other notes untouched.

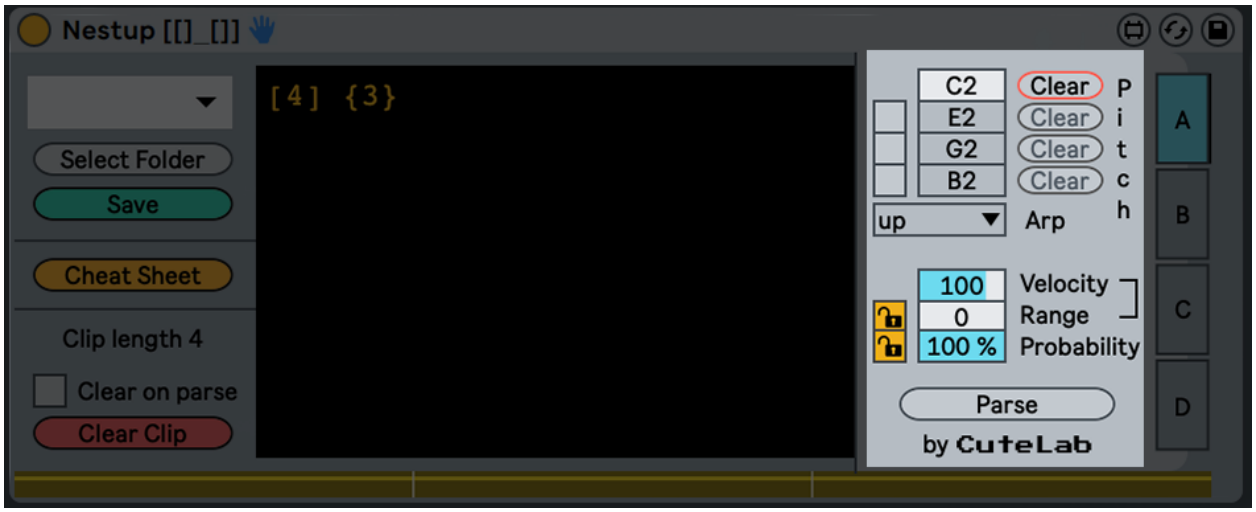
Pattern Entry

The center of the device is for Pattern Entry. This is where the magic happens. The Nestup Max for Live device takes expressions in the Nestup language and translates them into MIDI notes. What kind of MIDI notes depend in large part on what you write in this text box.

You don't have to be highly skilled in the Nestup language to start working with Nestup expressions. The pre-written patterns, accessed by the menu in the left part of the device, provide useful starting points for your own rhythms. If you want to dive into the Nestup language, you can look in the appendix, read the full documentation at our GitHub (https://github.com/cutelabnyc/nested-tuplets), or check out the the online demo at https://nestup.cutelab.nyc.

Once you've written a Nestup expression, you can press the **Parse** button in the bottom-right of the device, or press Shift+Return to parse the current expression. This will cause Nestup to write MIDI notes to the first clip slot in the current track. You'll also see a visualization in the bottom of the device.

Sequence Effects

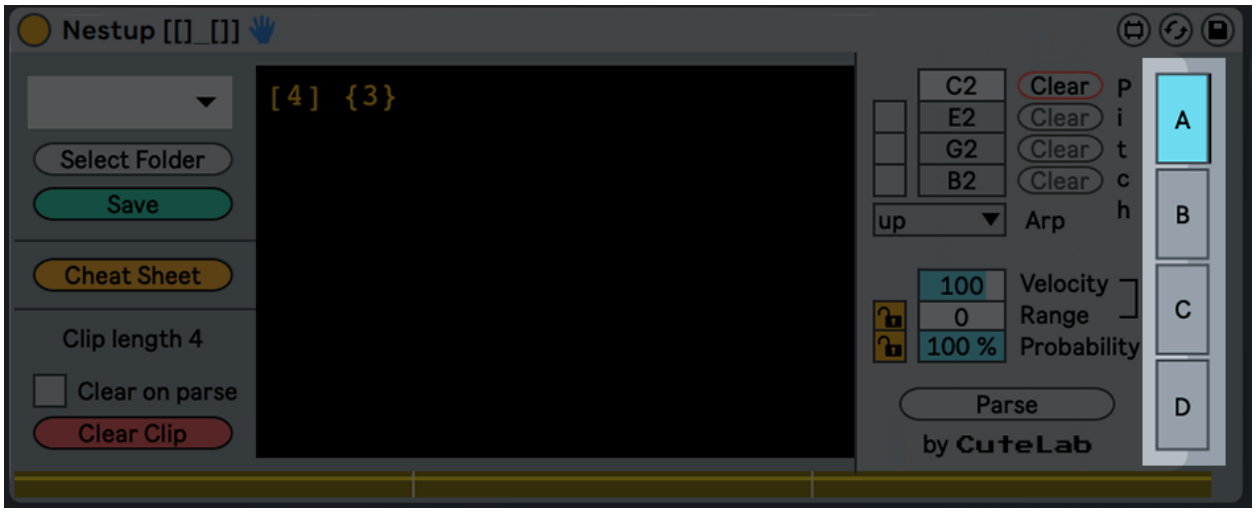


After generating MIDI notes, you can add additional effects with the controls in the right part of the device. These sequence effects let you change pitch, arpeggiate, and adjust the velocity and probability of MIDI notes.

The top half of these controls are devoted to pitch. Normally, the top pitch control determines the pitch of each note in the generated MIDI sequence. However, by activating the toggles on the left, you can add more pitches to the generated sequence. The menu at the bottom of the pitch section lets you decide how Nestup cycles between the active pitches. Finally, the clear buttons to the right lets you clear a row of pitches from the MIDI clip. It can be hard to know without looking at the clip whether there are any notes with the selected pitch, so the **Clear** button will change color when there are MIDI notes present in the selected pitch row.

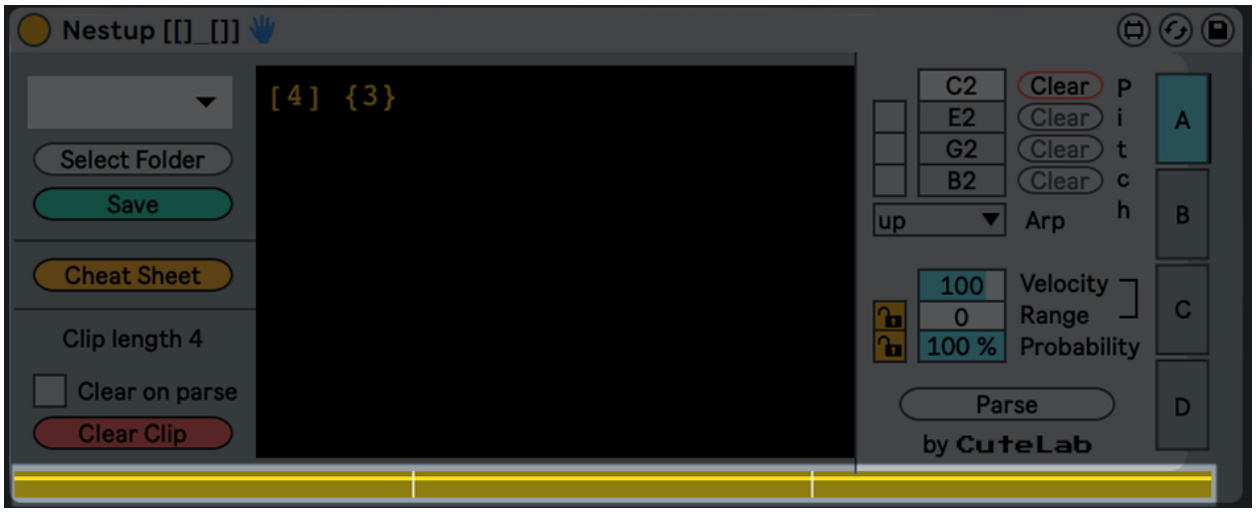
The controls in the bottom half of this section are for adjusting **Velocity** and **Probability**. The **Range** control is linked to **Velocity**, creating a range across which velocities will be distributed randomly. These controls can operate in either **locked or unlocked** mode. In locked mode, probability and velocity are fixed once the sequence is generated. Some notes may be deleted from the final sequence, depending on your probability setting. In unlocked mode, velocity range and probability are assigned to each note in the sequence on a per-note basis. Unlocked mode is only available in Live 11+.

Swap Tabs



On the far right of the device are the swap tabs. These let you jump quickly between different groups of settings. If you write a Nestup expression in tab A, and then move to a different tab, your expression will be restored when you come back to tab A. The same is true for your pitch, probability and velocity settings.

Visualizer



The visualizer at the bottom of the device gives you a preview of the sequence that Nestup has generated. This is just a preview, not a reflection of the notes in the MIDI clip. If you switch to a view of the MIDI clip and move or delete notes in that clip, the visualizer will not reflect these changes. The color has no real significance, although Nestup does try to give groups of notes the same color.

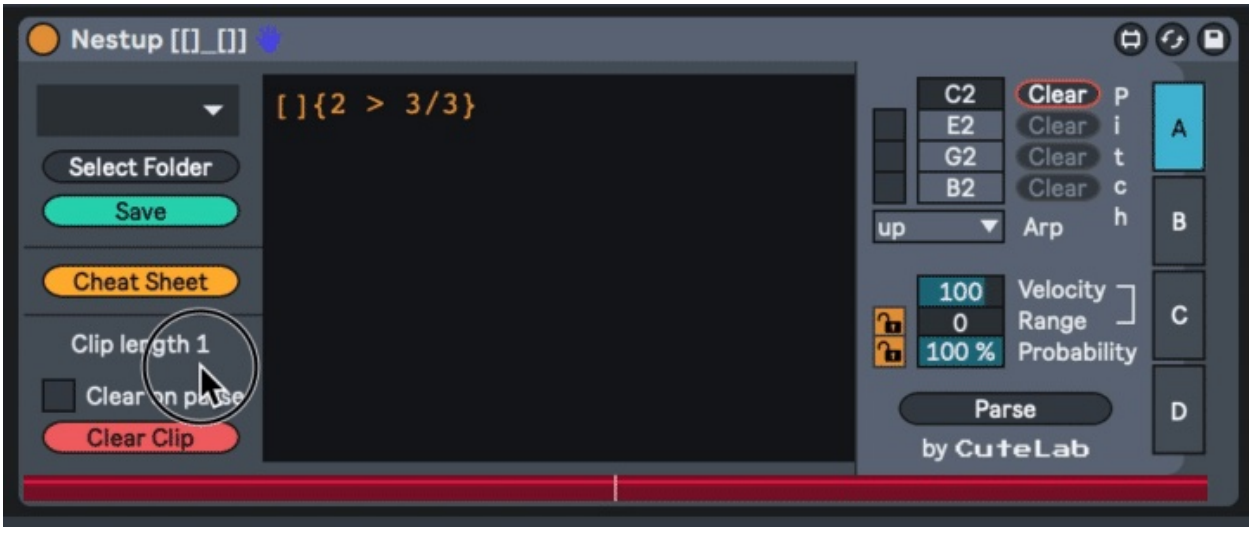
The brighter line of color inside of a note displayed in the visualizer will give you an indication of the velocity and range for that note.

Known Bugs + Caveats

Not to big it up too much, but Nestup is pushing the boundaries of what is currently possible in a Max for Live device. The code entry panel in the center of the device is actually an embedded local web page (Nestup does not require an internet connection). To our knowledge there are no other Max for Live devices like this, and it does create a couple of rough edges. These are bugs that will be fixed, eventually, but we don't have a timetable for when Ableton and Cycling '74 will be able to address them. They are not bugs that will stop you having fun with Nestup, but you will probably have to get used to them.

The main issue with Nestup is keyboard focus. If the device is not in focus (with the title bar of the device illuminated, for example), commands, such as ⌘+C or ⌘+V on Mac, will apply to the Live Session and not to the Nestup device. You may be able to type into the Nestup code box, but this issue will persist. You might try to copy text with ⌘+C, only to find that you've copied a Live track instead. Also, certain keys have an effect on the Live set as well. So you might type a "0" and find that you've disabled your device, or that you've muted your track by pressing "m", even though you're typing into the Nestup codebox.

To get around this, we recommend clicking the background of the Nestup device (for example, near the words "Clip length") before clicking inside the codebox. This should resolve most focus issues.



Troubleshooting

1. If the device looks like the image below, you need to update the bundled version of Max that comes with Live. Simply update your copy of Live 10 or Live 11 to the latest version.

2. If the code box disappears or draws incorrectly, first make sure you're using the latest version of Nestup, which you can always find on Gumroad.

3. If you're using the latest version of Nestup and the code box still disappears, try deleting and recreating the device.

4. If Nestup still won't draw correctly, contact us at cutelabnyc@gmail.com.

Appendix - Language Reference

This is intended to be a quick overview of the Nestup language. For a full breakdown of the whole language, visit the language homepage at https://github.com/cutelabnyc/nested-tuplets.

Containers

The core of the Nestup language is the container. You can make a container using square brackets, so a container like this:

[4]	

would be a container with a single note event lasting for four beats. Containers can contain other containers.

[[1] [2] [1]]	

In this case, the size of the parent container is the sum of the size of its children. A container can also have a fixed size, in which the size of its children determines their relative proportions. For example:

[4 [2] [3]]	

This container has size 4, and the two child containers divide that size proportionally (into 1 ⅗ beats and 2 ⅖ beats respectively). Containers are filled with a note event by default, but they can also be empty. An empty container is denoted with a single quotation mark before the size, if any. Empty containers can be useful for spacing.

[‘4]	An empty container of size 4
[[2] [‘1] [2]]	

Finally, a fixed or flexible container can have an overall scale. This scale can be an integer or fraction, which will be applied to the overall size of the container.

[[2,1/2] [2]]	

Here, the first child container has size 1, since it's fixed size of 2 is being multiplied by its scale of ½.

Subdivisions

If you want to give a container a number of evenly spaced child containers, you can specify any positive integer as a number of subdivisions.

```
[ 2 ] { 4 }
```

This is a container of size 2, evenly divided into four parts. Once the container is subdivided in this way, each subdivision can itself contain more subcontainers. This is accomplished by specifying a range, followed by a flexible container.

```
[ 2 ] { 4
  2 [ [] [] ]
}
```

This is a container of size 2, divided into 4 parts. There is a subcontainer with range 2, consisting of two equal child containers. A range has an index as well as a length. If the length is not specified, it defaults to 1. A range length can be useful for stretching a subcontainer across multiple subdivisions.

```
[ 2 ] { 4
  2:3 [ [] [] ]
}
```

Here the subcontainer with two equal parts has been stretched out over three subdivisions. Finally, there are a couple of syntax shortcuts when working with subdivisions. First, if a subdivision has a ranged container that itself has more subdivisions, the container can be omitted. So these two expressions are the same.

`[ 2 ] { 4 2:2 [] {3} }`	`[ 2 ] { 4 2:2 {3} }`	

Next, if the ranged subcontainer is simply a container with no children, the entire container can be omitted. This can be very useful when filling up empty subcontainers one event at a time. So a cumbersome expression like

```
[‘4] {16 1 [] 5 [] 10 []}
```

can instead be written as

```
[‘4] {16 1 5 10}
```

Rotations

Rotations are a powerful way to change an entire subdivided container all at once. Given a subdivided container, a rotation can shift every event in that container forwards in time or back. Events that are pushed past the beginning or end of the container will wrap around to the other side. A rotation is specified with a < or > character immediately following the subdivision number. The unit of a rotation is the same as the number of subdivisions. Here is the same container with different rotations:

`[ ] { 2 }`	
`[ ] { 2 > 1/4 }`	
`[ ] { 2 > 1/2 }`	
`[ ] { 2 > 3/4 }`	

Ties

Sometimes it’s desirable to link events together across two different containers. For example, if a triplet follows a quintuplet, you might want to link events across the two containers. Without creating a whole new container and doing a lot of math, it might be difficult to achieve the desired effect. In this situation, a tie can be useful. Compare the same two patterns, with and without a tie.

`[] {5} [] {3}`	
`[] {5} _ [] {3}`	

Ties can go between any two containers that are children of the same parent container.