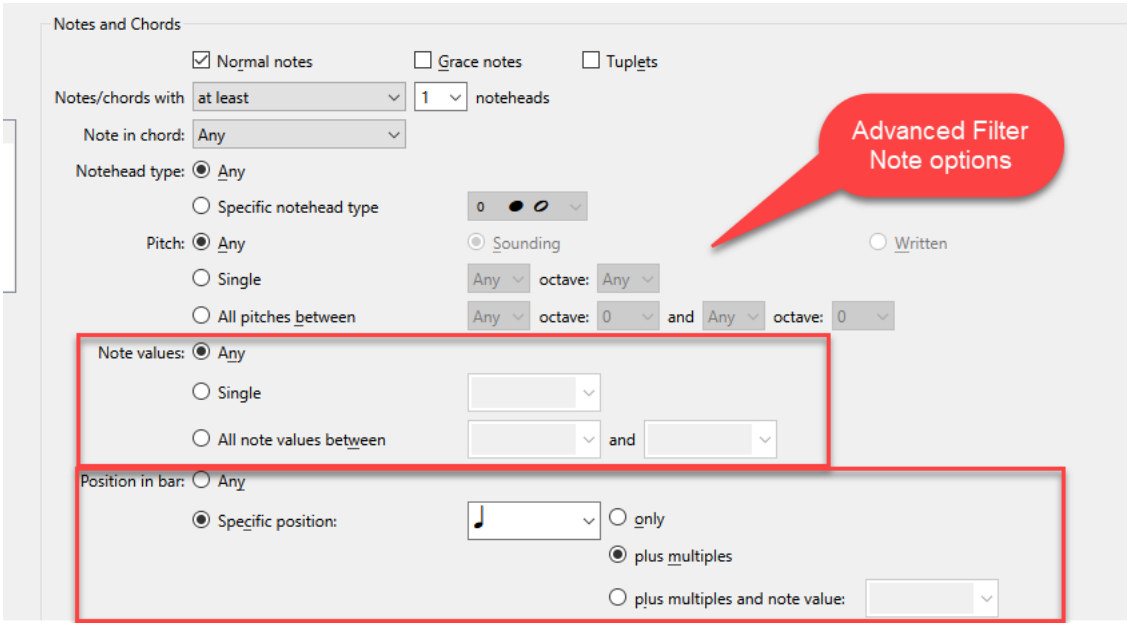
Filtering notes by duration or position or beat: Filter Notes By Duration, Filter Notes By Position, and Filter Notes By Beat

Bob Zawalich July 26, 2021

Three new downloadable plugins (**Filter Notes By Duration, Filter Notes By Position,** and **Filter Notes By Beat**) are being published to filter notes much in the same way that the Advanced Filter (AF) does.

One reason to do this is to allow a user to use a shortcut to filter notes in similar ways to what AF does, and to make it possible to do such filtering in macros using the **Execute Commands** plugin. Another reason is to focus the filtering process on a single type of filtering, rather than dealing with all the options available in the AF. I have always found the AF to be rather intimidating, and fairly often do the wrong thing.

I also wanted to have slightly different options for filtering compared to the AF. Here is a relevant chunk of the Advanced Filter dialog.

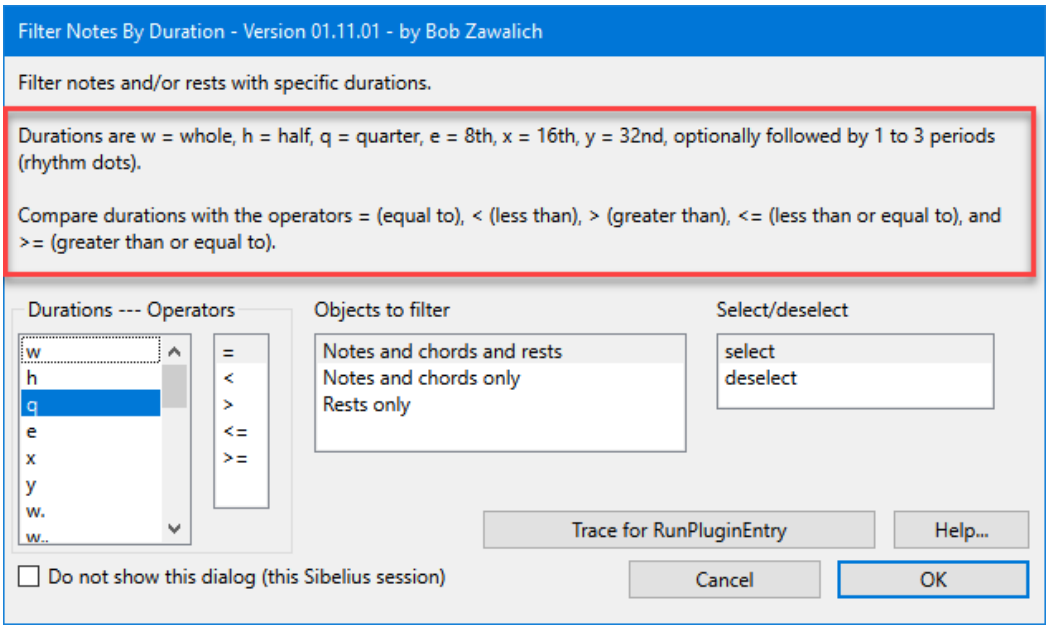


These plugins employ a new plugin structure that I call “Parent plugins”. While they can be used as other plugins are used, their design isolates the code that brings up the dialog and collect data from the dialog from the code that uses that data and processes the score.

This allows other, “Child”, plugins to call directly into the score processing code without needing to deal with the dialog. This is particularly useful for the **RunEntryPlugin_cu** command used by the **Execute Commands** plugin. These 3 plugins can all be run using **RunEntryPlugin_cu**. This is explained in some detail in Appendix 1 of this document.

Filter Notes By Duration

Filter Notes By Duration is a replacement for the Note values section of AF.



Like the AF, you can filter Notes chords and Rests, Notes and Chords, or Rests. Unlike the AF, the plugin only deals with filtering by duration, so there are fewer options to contend with.

Like the AF, you can select or deselect the matching notes.

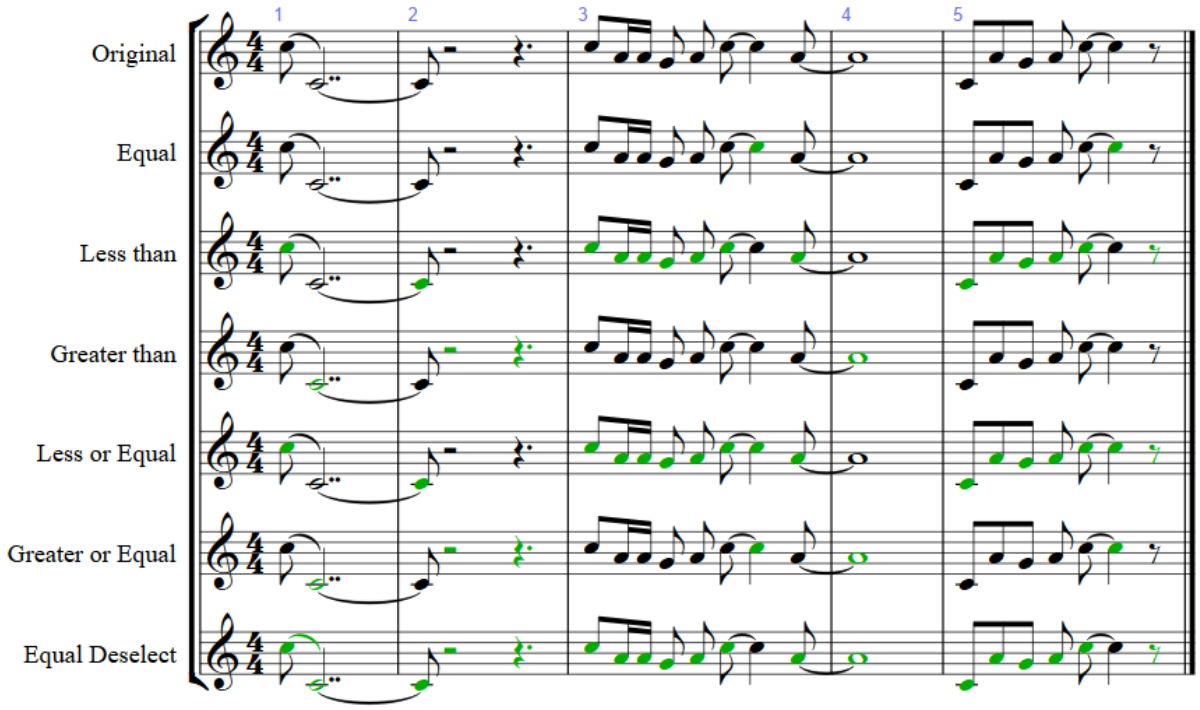
Like the AF, you can choose a duration between triple dotted whole notes to 32nd notes, though you need to specify durations with letter names instead of note graphics, and you cannot combine different durations, but can only choose from the list box. The list box shows the undotted durations first, followed by the dotted versions of the supported durations.

Unlike the AF, there is no option to filter a range of durations. There is a workaround for this that involves running the plugin twice, which I will explain later.

Unlike the AF, there are a set of comparison operators that can be applied to notes that match the filter. The AF effectively uses the "=" comparison, where notes are filtered if the note/rest duration matches the desired duration or range of durations. The plugin can match a duration , or accept a note if its duration is less than (<), greater than (>), less than or equal to (<=) or greater than or equal to (>=) the specified duration.

Here is an example of filtering notes using the various operators, and one example of using the Deselect option. I have colored the notes that would be filtered green.

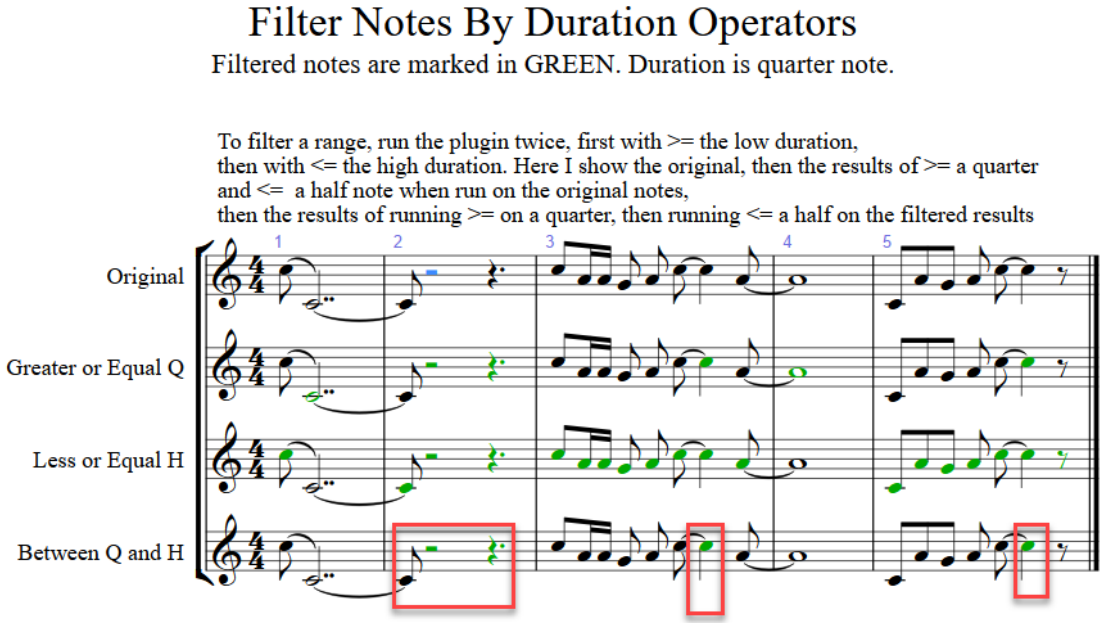
- Equal: only the 2 quarter notes and rests are selected
- Less than: 8th and 16th notes and rests are selected.
- Greater than: Dotted quarter, half, and whole notes and rests are selected.
- Less than or Equal: quarter notes are included with the Less than results.
- Greater than or Equal: quarter notes are included with the Greater than results.
- Equal with deselect: Everything except quarter notes and rests are selected.



Filtering a range of durations

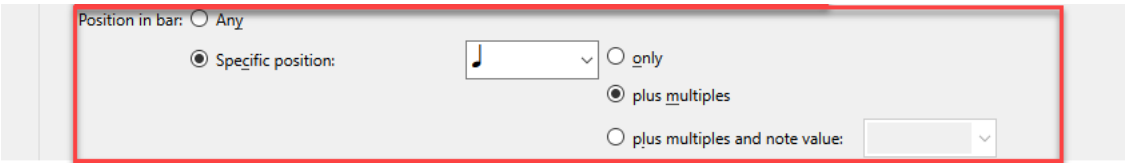
This requires running the plugin twice. First run it with the **Greater than or equal** operator using the smaller duration, then run it again on the resulting selection with the **Less than or equal** operator.
In this example the final staff shows the results of running >= Quarter, followed by <= Half. In the results, only quarter notes, a dotted quarter rest, and a half rest are filtered.

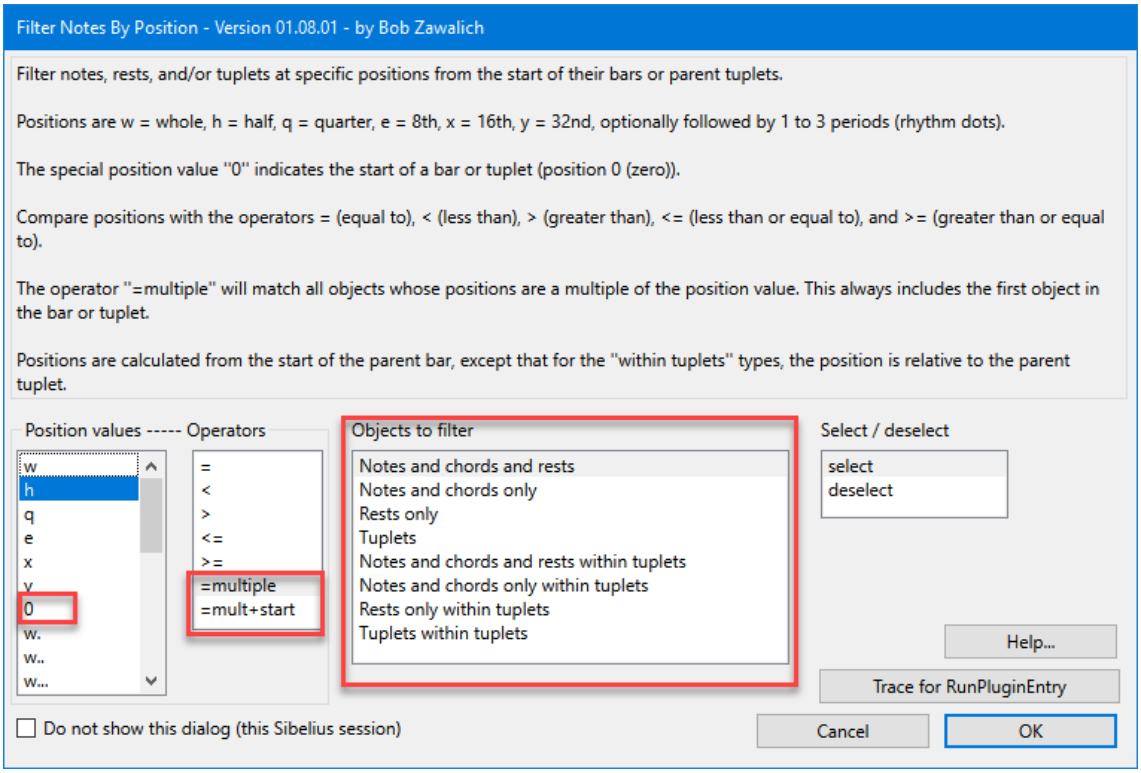
It is inconvenient to have to run the plugin twice, but if it is done in a macro, it is less of a problem, and removing the Range option greatly simplifies the setup for macros.



Filter Notes By Position

Filter Notes By Position is a replacement for the Position in bar section of AF.





It is very similar to **Filter Notes By Duration**, but the differences are significant. It reproduces the AF **Specific position** features, including 2 forms of **plus multiples**, but does not reproduce **plus multiples and note value**. I always found that too confusing to deal with, so I just left it out.

I will focus on some of the differences with **Filter Notes By Duration.**

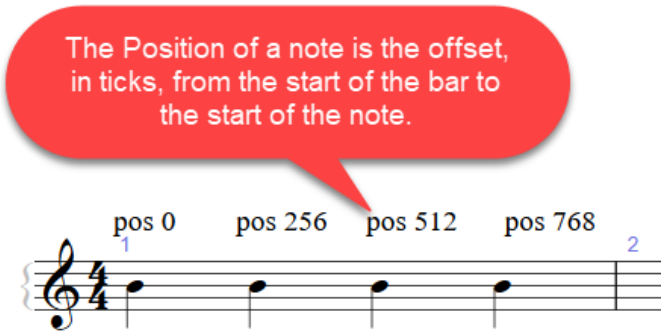
Positions and **Durations** are quite similar, and they use the same units, but they are not the same. The duration of a note specifies how long the note will be played, independent of the tempo. It is specified internally in rhythmic units called **ticks**, where a quarter note has 256 ticks. How long such a note is actually played in time is determined by the tempo (in beats per minute, and I will discuss beats later on), but the **Duration** value is important to Sibelius.

The **Position** of a note is the offset of the start of the note from the bar or tuplet where the note lives. (Here I will say **note** to mean single notes, intervals or chords, or rests, and generally if I refer to a bar, the same things apply to offsets in tuplets).

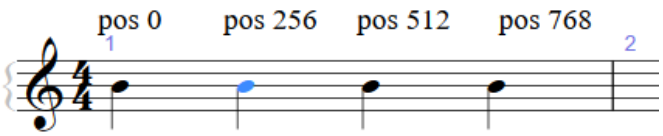
An Example:

If we have a 4/4 bar with 4-quarter notes in it, the position of the first note is 0. The second note is at position 256, the 3rd at 512, and the fourth note at 768. These are all multiples of 256. The first note will start playing at the start of the bar, and it plays for 256 ticks, (ticks 0 – 255), the second plays from 256 to 511, the third from

512 to 767, and the 4th note plays from 767 to 1023, which is the last available tick in a 4/4 bar (since the numbering starts at 0, tick 1023 is actually the 1024th tick in the bar).



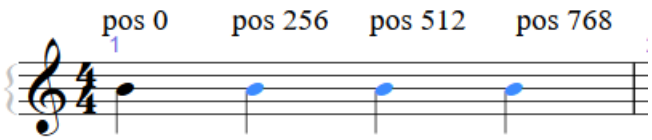
If we want to filter the note at position 256, we specify the position q, for quarter note. This is the note that is a quarter note from the start of the bar. If I fully select the bar above, and run the plugin using q, =, Notes and chords and rests, select, I will see this.



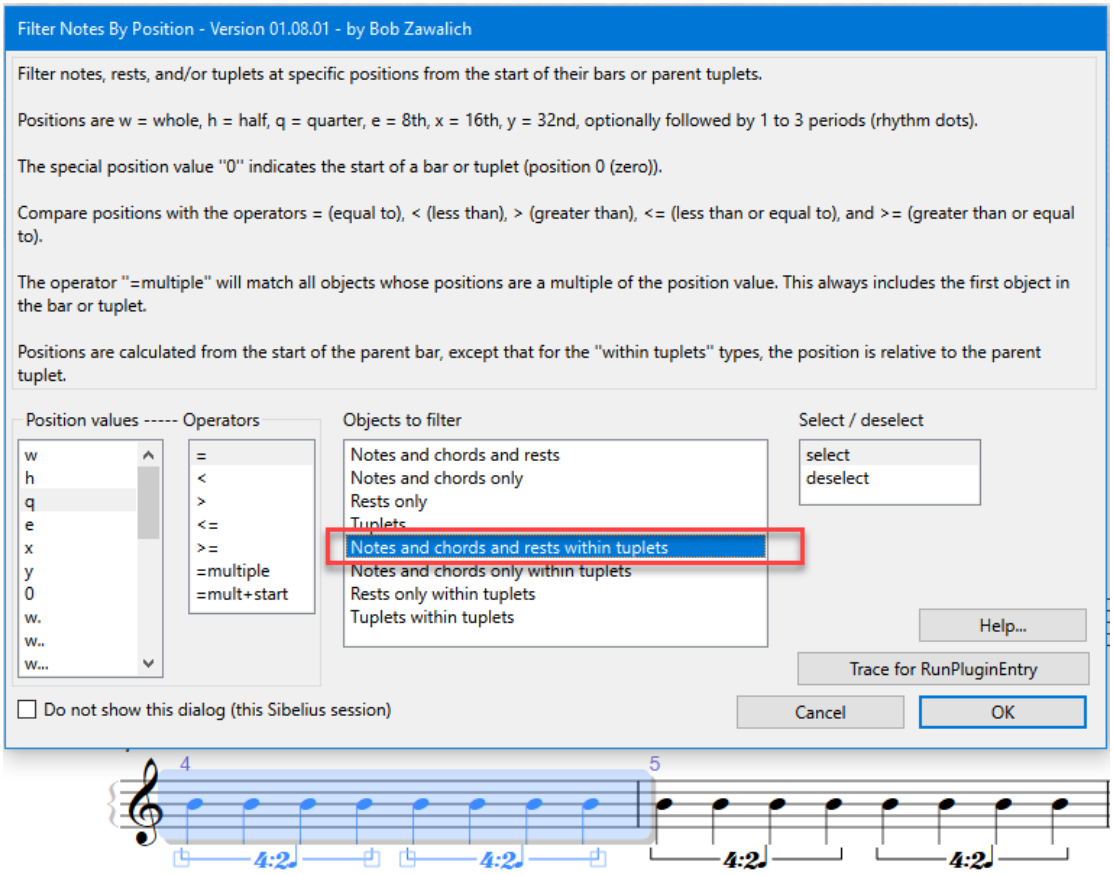
What if I want the position at the start of the bar? In that case I use the special position value 0 (zero).

Filtering on a position works pretty much as I would expect it to. It processes each bar in the selection separately, and if there are bars with no notes in a desired position (like when you have only a bar rest and filter for position q), nothing will be selected in the bar. If no notes are matched at all, the selection will be unchanged.

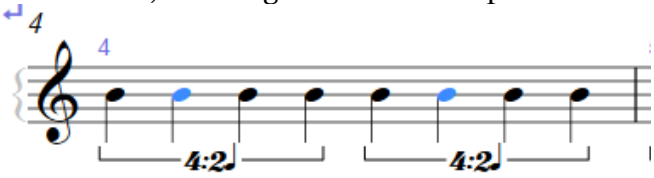
I can use the different operators to modify which notes get filtered, In the above example if I switched the operator to “>=” instead of “=”, I will see



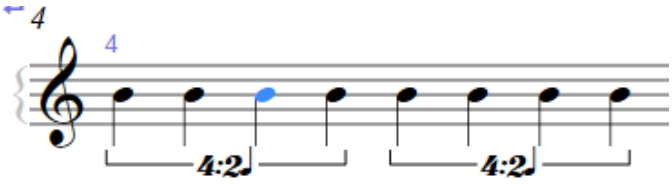
I also have a wider range of objects to filter than in **Filter Notes By Duration**. I can filter tuplets, and I can also filter objects relative to a parent tuplet instead of a bar. In such cases, the plugin will process each tuplet in each bar separately. If I have 2 tuplets as shown below, and ask to filter notes at position q (quarter note) within tuplets,



I see this result, selecting the note whose position is a quarter note from the start of each tuplet:



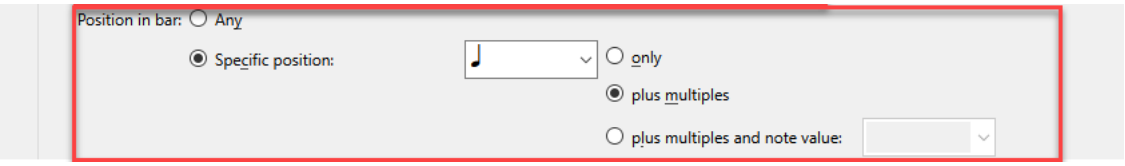
If I used the normal Notes and chords and rests on the same selection, it would treat the position of each note as relative to the bar, and the result would be different and probably unexpected. I will show you the math for this in an appendix, but just be aware the positions within bar and tuplets are different.



=multiple and =mult+start

I don't know about you but I pretty much never use the **multiples** options in the AF, because I always get unexpected results. I was leery about trying this but I figured people would want it. So what does **plus multiples** mean, and how does Sibelius implement it?

As I said, a position is an offset in ticks from the start of a bar. Let's leave off tuplets here.
A quarter note is 256 ticks. Let's see the AF dialog again.



The Reference says:

> * Position in bar: by default, filters notes at Any rhythmic position, but optionally filters notes and chords at a Specific position after the start of the bar.
>
> Leave the Specific position box empty to filter just notes/rests at the start of bars.
> If you choose quarter note (crotchet), only notes that are one quarter note into the bar (i.e. begin on the second quarter note) will be filtered.
>
> If you also choose Plus multiples, every note on a quarter note (crotchet) beat is filtered.
>
> If you instead choose Plus multiple and note value, every note on a quarter note beat *plus* multiples of the specified duration is filtered.

It is really not obvious to me how to specify that you want to filter a note at the start of the bar (in fact what looks like a list box is really a "combo" box which acts both as an edit box and a list box, so you choose Specific position but erase the note symbol), but I get around that by providing a position 0 (zero) that indicates the start of a bar or tuplet.

So leaving off position 0 for now, what does a multiple of, say, an 8th note get you?

I would think it would filter the first 8th note, then the 2nd 8th, and the 3rd in the bar. So if I have a bar full of 16th notes and applied the filter, I would get the 3rd (which is the first note at offset 8th note), 5th, 7th,... notes selected. Instead I see this:

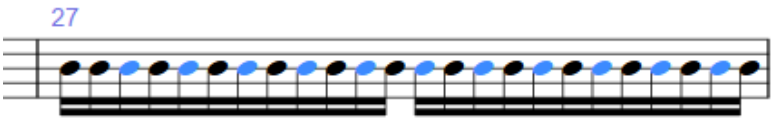
which is almost what I expected, but the first note in the bar is also selected. Is that what

> If you also choose Plus multiples, every note on a quarter note (crotchet) beat is filtered.

means? (Not to mention that I had to go to the Reference to find that). Maybe.

So what it does is treat the first note in the bar as always matching, which is OK I guess, but I never think of multiples as going backwards, which seems to me to be what this is doing.

So in **Filter Notes by Position**, the **=multiple** operator selects the first matching note, and then any **following** notes that are multiples of the position. So we get this, excluding the first note:



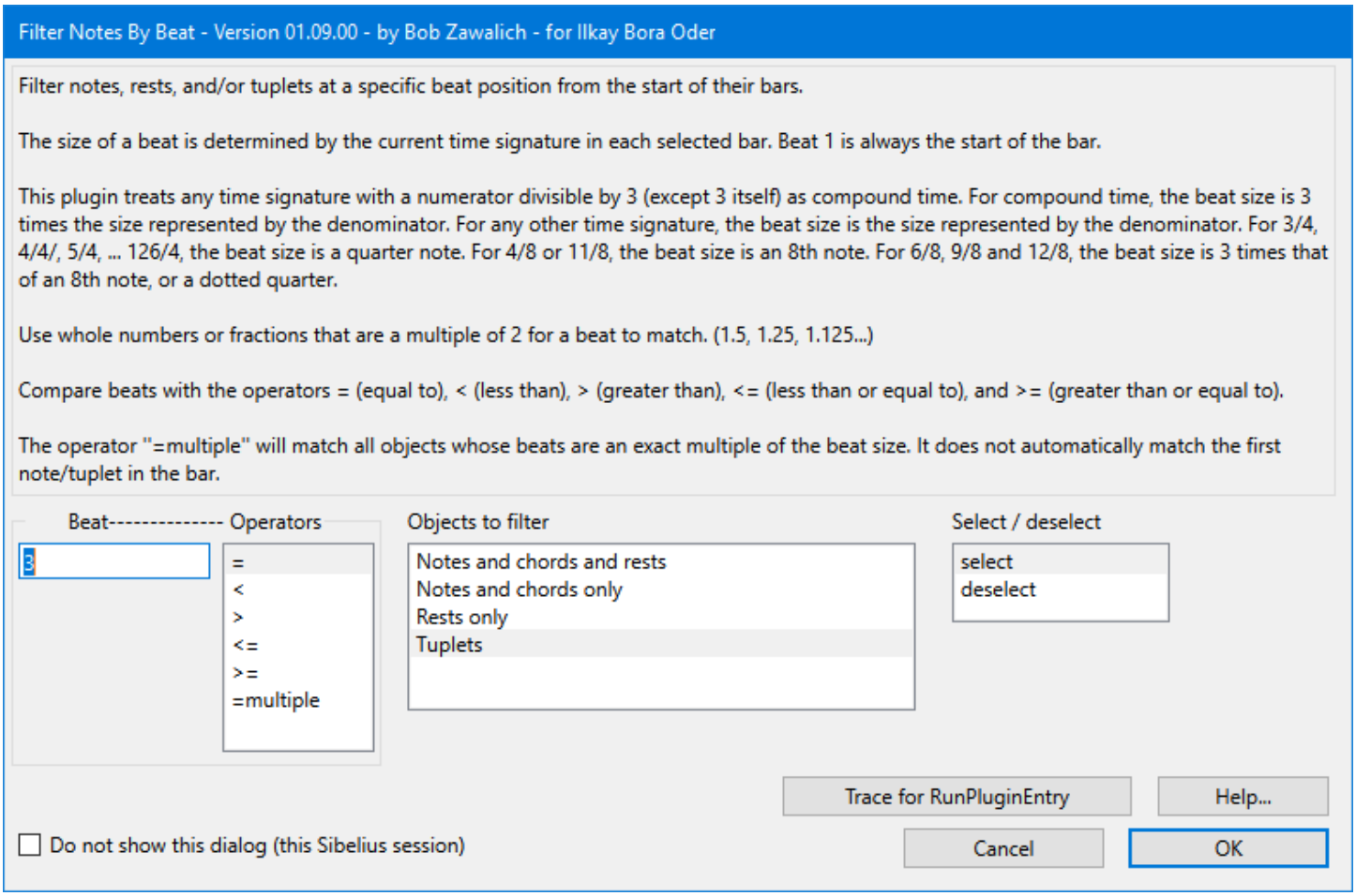
But what if you do want the first note selected as well? Use the **=mult+start** operator instead.

And how about multiples of the position 0 at the start of the bar? What is a multiple of zero? Should it select the entire bar? Every note in the universe?

I decided arbitrarily that in that case it should just select the first note in the bar/tuplet, because it is easy to select all or nothing, and selecting just the first note could be useful (though you can get it by just using the = operator).

Filter Notes By Beat

This is not in the AF, but it is a variant on **Filter Notes By Position**.



This plugin is dedicated to **Ilkay Bora Oder**, who suggested it, along with many other good suggestions, and who was my main cmdutils tester despite many terrible crashes. cmdutils would not be the same without his input, nor would this plugin exist!

If you are trying this out, I strongly suggest running the shipping **Number Beats** plugin on your test material, as I did in the examples below. It makes it a lot easier to see if the beat you wanted is the one that got filtered, especially if you are using complex time signatures, rhythms, or tuplets.

Instead of a list of position values, there is an edit box in which you can type a beat value, which must be a positive number greater than or equal to 1. Beat 1 represents position 0 in a bar.

Due to rounding it is unlikely that Notes inside tuplets will get matches, and the beat value will work best if it is either an integer (1, 2, 3...) or if a fraction, it is a value divisible by 2, like 1/2 (.5), 1/4 (.25), 1/8 (.125, etc.).

This plugin does not have the Tuplet-relative options that **Filter Notes By Position** does, and the **=multiple** option does not select the first note in the bar.

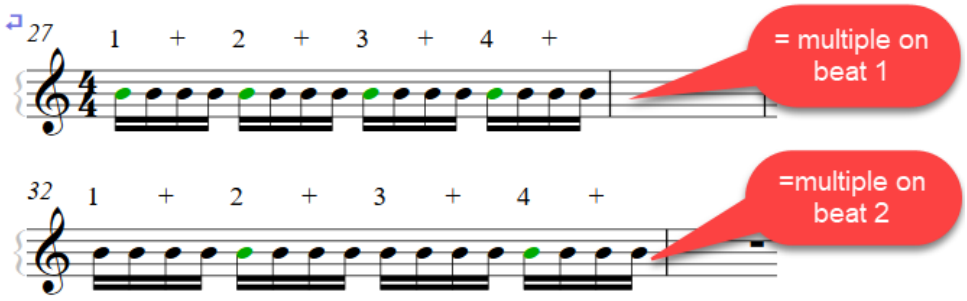
To determine the size of a beat, the plugin treats any time signature with a numerator divisible by 3 (except 3 itself) as compound time. Any other time signatures a treated as simple time for size calculation.

For compound time, the beat size is 3 times the size represented by the denominator. For any other time signature, the beat size is the size represented by the denominator.

For 3/4, 4/4/, 5/4, ... 126/4, the beat size is a quarter note. For 4/8 or 11/8, the beat size is an 8th note. For 6/8, 9/8 and 12/8, the beat size is 3 times that of an 8th note, or a dotted quarter.

In the Sibelius status bar, the beat numbers shown by 6/4 do not treat it as compound time (the beat is a quarter note rather than a dotted half), though in playback, the Transport window shows a 6/4 beat size of a dotted half. This is the only difference I am aware of in the beat size shown in this plugin and the status bar.

=multiple for beat 1 will select every note starting on the beat. For other beat values, the multiples will always be later than the first matching value.



In the beat 2 example, I think it would be weird to select the first beat in the bar, so that multiples of beat 2 would be beats 1, 2, and 4. Continuing that for larger bars, we would get beats 6, 8, 10, etc. Selecting beat 1 in that sequence just seems weird to me, so I am not going to do it.

There really is not a way to get the same effect with multiple positions in **Filter Notes By Positions** as I do here with multiple beats.

Appendix 1: Using Execute Commands, cmdutils and the RunPluginEntry_cu command

These plugins were designed from the start to be able to be called from Command Macros and Command Plugins in **Execute Commands**, and this section applies to all 3 plugins.

You do not need to use the plugins this way – they will run fine as "normal" plugins and you can hide the dialog if you want, but their use in macros was an integral part of their design.

The plugins are structured internally as "Parent" plugins, so that the dialog code and its data are separate from the code that does the filtering. When one of these plugins is run, as from a menu or shortcut, the code leading up to the dialog is run, then all the options shown in the dialog are stored into a data structure called a Dictionary, and that Dictionary is passed to a routine called **API_ProcessSelection**, which then does the filtering.

In most plugins the variables from the dialog are used throughout the code, but these plugins are designed so that the processing code only sees what is in the Dictionary.

Other plugins ("child plugins") can put the same dialog setting values into a properly designed Dictionary, and then call **API_ProcessSelection** directly, and the results will be the same as if you run the parent plugin with those dialog settings, but without showing a dialog.

The RunPluginEntry_cu command

RunPluginEntry_cu is a command in the cmdutils library. It is effectively a universal child plugin that can call any parent plugin that uses the model of separation of dialog data. The command has a set of parameters that look like this:

> RunPluginEntry_cu(<plugin name>, <plugin entry point>, <data name *1*>, <data value 1>, <data name *2*>, <data value 2>, …<data name *n*>, <data value n>)

This model allows me or any other developer to extend what is available to a macro by writing plugins that can be called using **RunPluginEntry_cu/**.

You usually will not need to know the details or the **RunPluginEntry_cu** command line, because these plugins will generate and trace these command lines for you, but it might be useful if things go wrong.

The parameters to **RunPluginEntry_cu** are the plugin name and entry point, followed by pairs of (name, value) text, usually representing data from a dialog. The parameters are separated by commas and optional spaces.

We could call **Filter Notes By Duration** via **RunPluginEntry_cu** with the dialog options shown in the screenshot above using this command line:

> RunPluginEntry_cu(FilterNotesByDuration.plg, API_ProcessSelection, str_Duration, q, str_Operator, =, str_Type, Notes and chords and rests, str_Action, select)

Here the parameters are:

<plugin name>: FilterNotesByDuration.plg
<plugin entry point>: API_ProcessSelection

<data name *1*>: str_Duration
<data value 1>: q

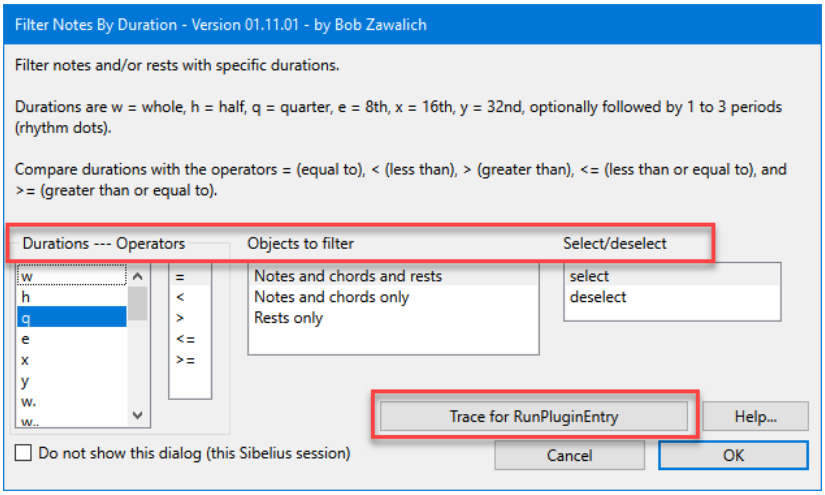
<data name *2*>: str_Operator
<data value 2>: =

<data name *3*> : str_Type
<data value *3*>: Notes and chords and rests

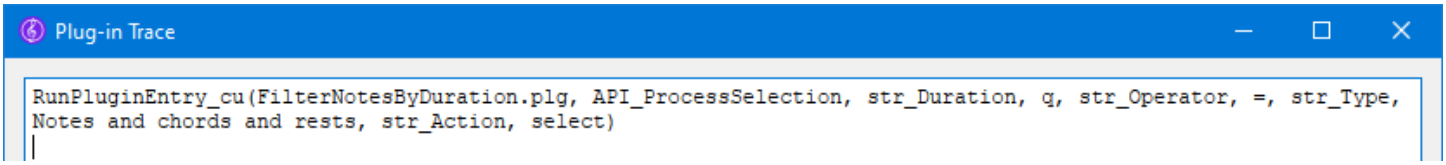
<data name *4*>: str_Action
<data name *4*>: select

The first 2 parameters are always present, and the tell the command which plugin and which routine inside the plugin should be called. The number, names, and values of the data (name, value) pairs will change with every plugin.

In this example, you can see that the data corresponds to the various list boxes in the dialog for **Filter Notes By Duration**. Here is that dialog again.



For these plugins, you don't really have to worry about correctly typing the **RunPluginEntry_cu** command line, because they have a button on their dialogs labeled "**Trace for RunPluginEntry**". You can choose the dialog options you want, and click the button, and the appropriate text will appear in the plugin trace window. For the dialog above you would see



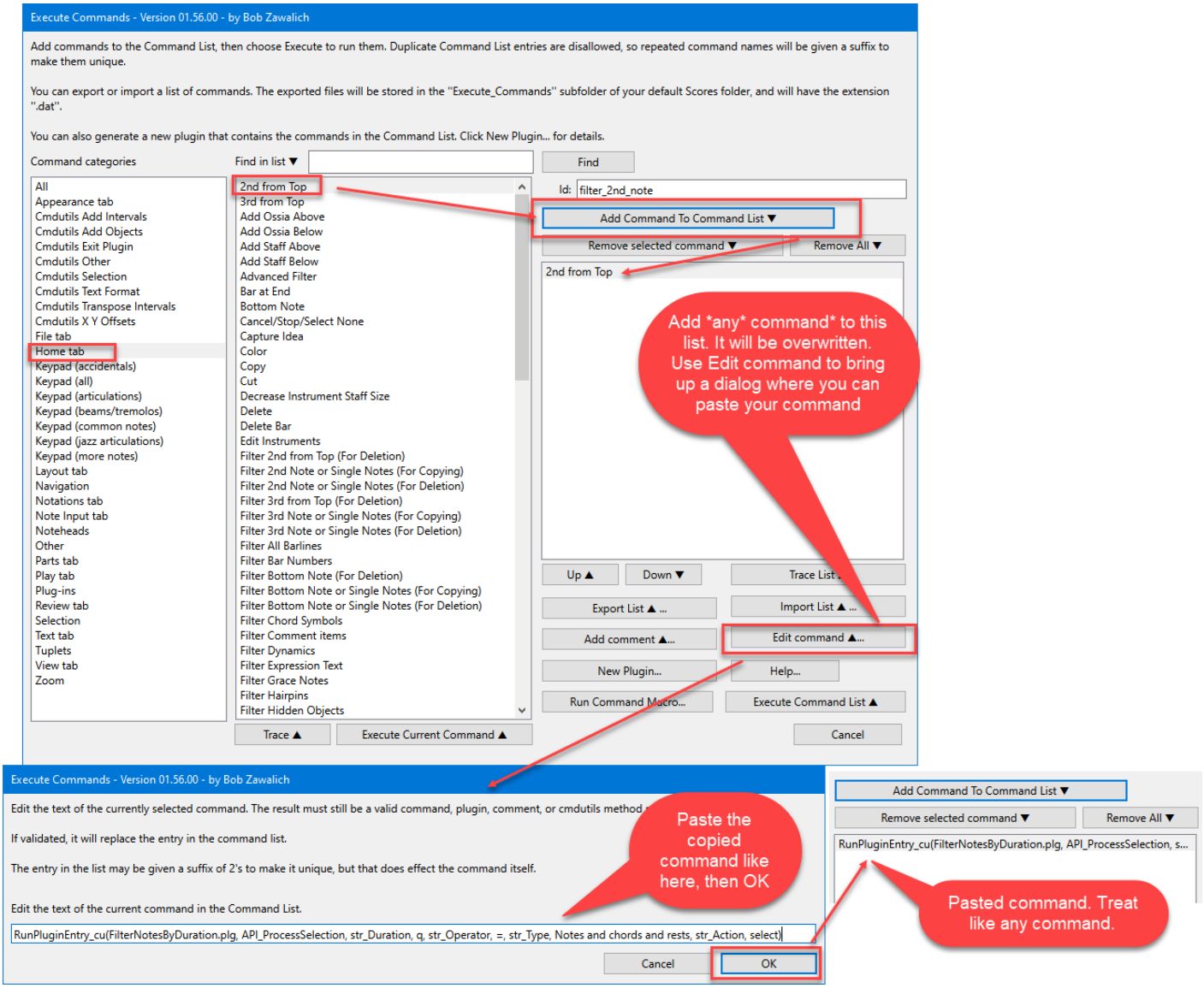
One way to use this in **Execute Commands** is to use **Trace for RunPluginEntry**, then select and copy the text. Try to select only up to the closing parenthesis when copying.

(Hold onto your hat for a moment here!)

Now run **Execute Commands**, and add a command (any will do) to the **Command List**. In the **Command List,** choose that command, and click on **Edit Command**. When that dialog comes up, paste the text you has copied earlier into the edit box, overwriting the command. Click OK, and the pasted text will now be in the command list. Here is a non-animated animation of that process.

- First we use **Add Command To Command List** to add any old command to the command list – since it will be overwritten it does not matter which command you choose.
- Now click on Edit Command. You will see a dialog box with your command (**2nd from top**, in this example) selected in its edit box.
- Use ctrl/cmd+v to paste your copied **RunPluginEntry** command like into the edit box. With luck all the text will be visible in the edit box, but the ends of very long strings will be scrolled off the edit box.

- Press Enter or click OK, and you will return to **Execute Commands** and you will see the start of the command in the command list. Hover your mouse over the line to see the full text. Use **Edit command** if you need to inspect or edit the whole line.



You can instead paste the command line into a text editor, and save the text as a text will with a .dat extension. Save in in the **Execute_Commands** subfolder of your default **Scores** folder, which is where **Execute Commands** saves macro files. You can then use **Import list** to read the command into the **Command List**.

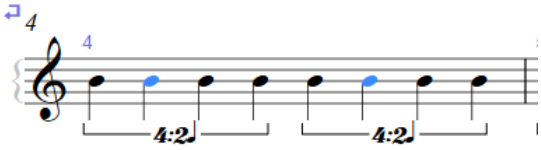
At this point it will behave like any command. Execute it in **Execute Commands**, or export the list to a macro file (I recommend always doing this with these commands, because they will not appear with their parameters in the **Command Category** lists), or use **New Plugin** to generate a plugin from it, or mix it with other commands.

When that command is run, the desired filter will be made.

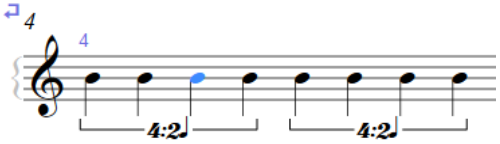
While the default **RunPluginEntry_cu** command is in the **Execute Commands** command list in the **Cmdutils Other** category, the modified versions you have here are only available in **dat** files or plugins. I recommend always exporting them into a **dat** file using **Export list** so you will have it for copying or editing later.

Appendix 2: Calculating positions in tuplets

Earlier I showed an example of using **Filter Notes By Positions** to filter for notes at position q (quarter note) within a bar, rather than within a tuplet, when I had notes in a tuplet. If I filtered for notes in a tuplet I would get



But in the bar, I got



Why?

Sibelius and plugins can look at 2 different positions for NoteRest object. The normal position in a bar is the distance from the start of the bar, in units relative to the bar. If the NoteRest is in a tuplet, there is another position value called **PositionInTuplet**, and in that case, the position is scaled by the ratio of the tuplet, in this case 4:2.

For the first tuplet, the "nominal" positions of the notes (if they were not in a tuplet) would be 0, 256, 512, and 768.

Since they **are** in a tuplet, the "real", or "played" position (and duration) of the notes is divided by the tuplet ratio 4/2. This is the same as multiplying the values by the inverse of the ratio 2/4, which is the same as ½.

Multiplying each position by ½ we get 0, 128, 256, 384. The second tuplet starts at position 512 from the start of the bar, and its adjusted positions are 512, 640, 768, and 896.

The plugin will also see the adjusted position in this case, so only the note at position 256, the 3rd note in the first tuplet, will be filtered.

Had these been 3:2 triplets, we would not find a match at all. The positions for 2-3:2 triplets of quarter notes would be 0, 170 and 341 for the first, and 512, 682, and 852 for the second triplet

because all the positions have to be multiplied by the tuplet ratio inverse 2/3. And there is a lot of rounding or truncation going on here. 256 x (2/3) is actually 170.666666…

The 0 and 512 values are the only really reliable values, and the others are all approximations.

Where possible I try to deal with positions and durations in tuplets in "tuplet units", rather than "bar units".