Tutorial: Execute Commands and cmdutils

Understanding and controlling the Selection

Bob Zawalich June 4, 2021

When you run a sequence of commands it is more likely to be successful if you understand what the selection is both before and after each command.

Most Sibelius Commands do not change the selection, but some, including any **Filter** commands, will change the selection so that only the filtered objects are selected. Plugins very often change the selection, and they often require a certain type of selection, such as a passage selection, when they start.

You want to be sure that the following command can work with the selection that the previous command produces.

There are some Sibelius Commands that allow you to control the selection, but most are not available to macros and plugins. The plugin **cmdutils** provides additional commands that will save and restore a selection, recreate a passage selection from selected objects, and perform other selection manipulations.

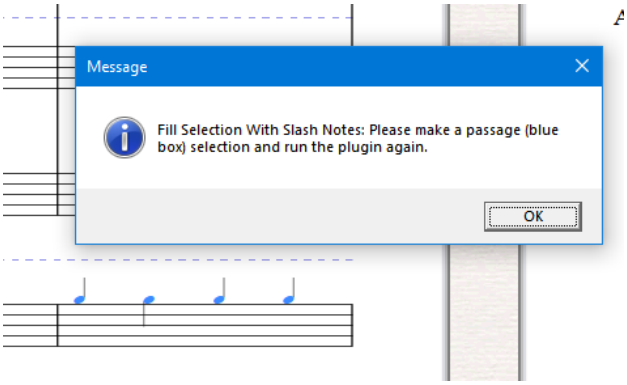
I would like to show some of the selection commands that are available in **cmdutils**, explain what they do, and show how they can be used in macros and plugins.

An example of a sequence that is thwarted due to selection changes

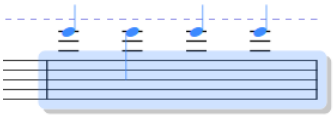
My friend John Hinchey sent me this sequence of commands, which was his attempt to recreate the function of the discontinued plugin **Make Pitches Constant Drums**.

Move Pitches To Transposed Mid Line (Plug-in 559)
Notehead 0
Add interval 6th above
Bottom Note
Delete
Cue size (on/off)
Fill Selection With Slash Notes (Plug-in 211)

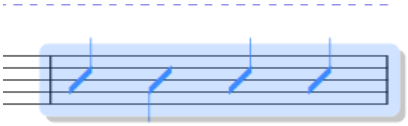
It seemed to be going fine, but then he saw this error message, and was wondering why.



I examined the commands, and ran them one at a time starting from this selection. I actually added an ExitPlugin_cu command to the Command List, and slid it up between the commands so I could control where the plugin stopped and see the state of the score after each command ran.

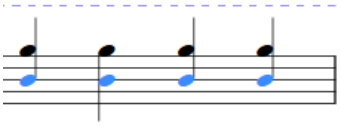


Move Pitches To Transposed Mid Line (Plug-in 559) is a plugin call. It requires a passage selection, so we made sure the starting selection was a passage. I ran it with the default settings, which changed the notehead style to a beat with a stem. At the end I had this selection:

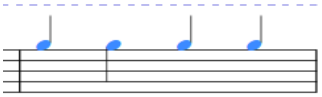


Notehead 0 restored the notehead styles to normal.
Add interval 6th above added a note a 6th above each selected note, and now we had this:

Bottom Note filtered the bottom note of each selected chord (the original notes), and now we had this, will is a collection of selected notes:

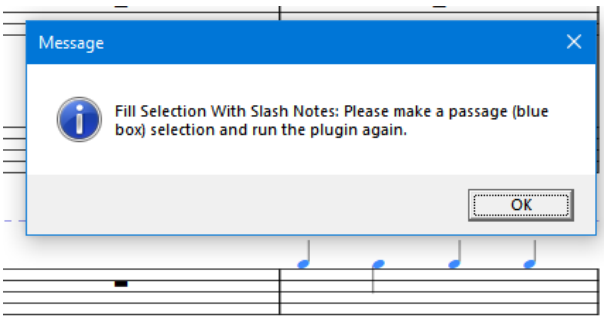


Delete removed the original bottom notes, and left us with the new notes, which were now selected



Cue size made them cue-sized, and they were still selected:

Now we ran **Fill Selection With Slash Notes (Plug-in 211),**and got the error message



Why? Because **Fill Selection With Slash Notes,** like **Move Pitches To Transposed Mid Line,** wanted a passage selection when it started up, and we did not have a passage selection anymore. Once we ran **Bottom Note** we just had individual selected notes.

There was no easy way to fix this at the time, other than creating a plugin from the commands and adding some additional code, but we can do it now by adding commands to save and restore the selection at appropriate times. Here is the original code again:

Move Pitches To Transposed Mid Line (Plug-in 559)
Notehead 0
Add interval 6th above
Bottom Note
Delete
Cue size (on/off)
Fill Selection With Slash Notes (Plug-in 211)

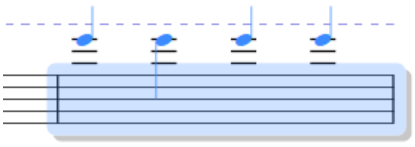
We know we have a passage select at the start, or **Move Pitches To Transposed Mid Line** would not work, and we know we need a passage selection for **Fill Selection With Slash Notes** to work. We also need a filtered non-passage selection for **Delete** to work.

We now have access to these commands, and can use them here

SaveSelection_cu ()
- Saves the current selection so it can be restores later. Most useful for a passage selection

RestoreSelection_cu ()
- Restores the most recent selection saved by SaveSelection

We can add **SaveSelection_cu** at the start, before we even call **Move Pitches To Transposed Mid Line,** and this is what we will save away:



The concept of what a selection actually contains is tricky, but for our purposes, assume it saves the "outline" of the selection. I can trace the outline in a plugin and it tells me I am selecting all of bar 3 in staff 9. (Trust me on this one).

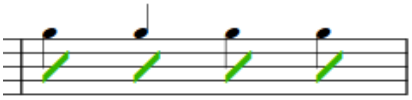
TopStaff = 9
BottomStaff = 9
FirstBarNumber = 3
LastBarNumber = 3
FirstBarSr = -1
LastBarSr = 32512

I can save the selection at the very start, and then restore it just before I call Fill Selection With Slash Notes.

SaveSelection_cu()
Move Pitches To Transposed Mid Line (Plug-in 559)
Notehead 0
Add interval 6th above
Bottom Note
Delete
Cue size (on/off)
RestoreSelection_cu()
Fill Selection With Slash Notes (Plug-in 211)

Now, before I call **Fill Selection With Slash Notes** the selection will be this passage selection

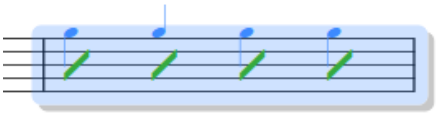
And the plugin will run, producing this:



Which is what John had wanted in the first place. Note though, that at the end, we again do not have a passage selection. **Fill Selection With Slash Notes** very kindly selected the added slash notes (in green voice 2) so we can see what was changed. This may be what we want, but suppose we really wanted a passage selection so we could run another plugin on the selection. We could just add another **RestoreSelection_cu()** call at the end,

SaveSelection_cu()
Move Pitches To Transposed Mid Line (Plug-in 559)
Notehead 0
Add interval 6th above
Bottom Note
Delete
Cue size (on/off)
RestoreSelection_cu()
Fill Selection With Slash Notes (Plug-in 211)
RestoreSelection_cu()

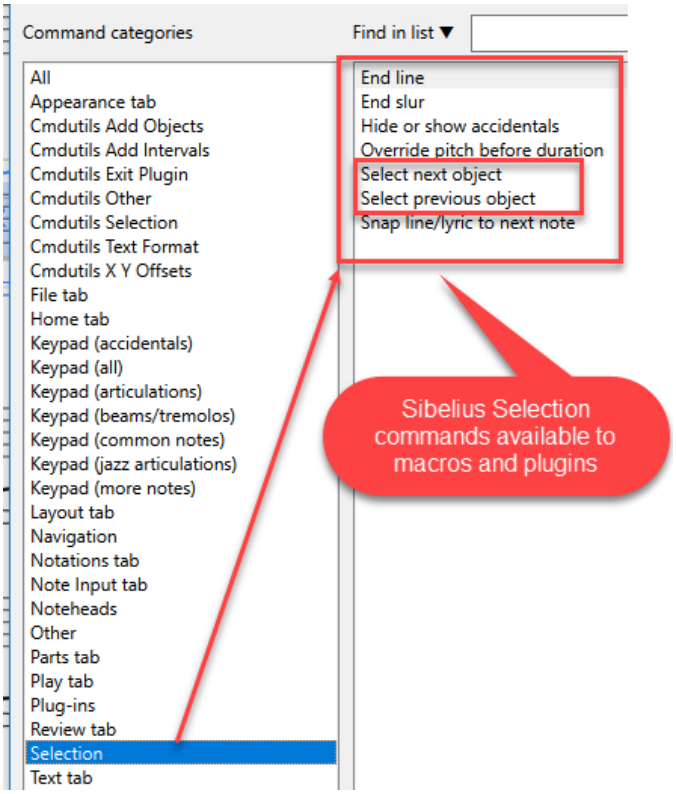
and we would have this:



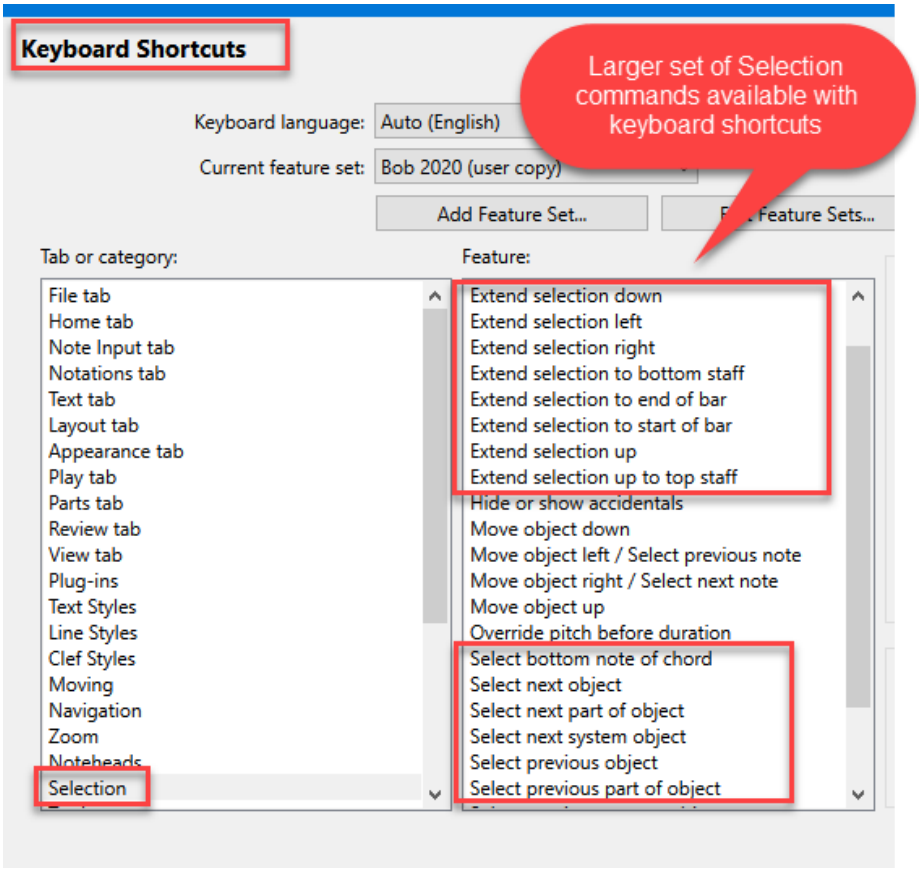
I note that if you had greatly altered what was in the selection such that the original selection did not make sense with the new objects it needed to select, then it would not be such a happy ending. But I find that there are many places where using **SaveSelection** and **RestoreSelection** appropriately will let you continue. I often use the same technique in ManuScript plugins as well.

I will now discuss some **Selection** commands that I have added to the **cmdutils** plugin that can be used in Command Macros and Command Plugins.

These are the Sibelius Selection Commands that are available to Command Search, Macros and Plugins as of Sibelius Ultimate 2021.2



There is a larger set of Selection commands available to Sibelius users via keyboard shortcuts



*These are the Selection commands available in **Execute Commands** in the **Cmdutils Selection** category when **cmdutils** is installed, including our friends **RestoreSelection_cu()** and **SaveSelection_cu()***

ContractSelection_Down_cu()
ContractSelection_Left_cu()
ContractSelection_Right_cu()
ContractSelection_Up_cu()

DeleteSelection_cu()

ExtendSelection_Down_cu()
ExtendSelection_FullBar_Left_cu()
ExtendSelection_FullBar_LeftRight_cu()
ExtendSelection_FullBar_Right_cu()
ExtendSelection_Left_cu()
ExtendSelection_Pages_cu()
ExtendSelection_Right_cu()
ExtendSelection_Up_cu()

FilterAllSelected_cu()

GoToFirstBar_cu()
GoToFirstObject_cu()
GoToFirstObject_SystemOK_cu()

GoToLastBar_cu()
GoToLastObject_cu()
GoToLastObject_SystemOK_cu()

GoToNextBar_cu()
GoToNextPage_cu()
GoToPreviousBar_cu()
GoToPreviousPage_cu()

MakePassageSelection_cu()
MakeSystemPassageSelection_cu()

RestoreSelection_cu()
SaveSelection_cu()

Select_All_NonPassage_cu()
Select_All_NonPassage_System_cu()
Select_All_Passage_cu()
Select_All_Passage_System_cu()

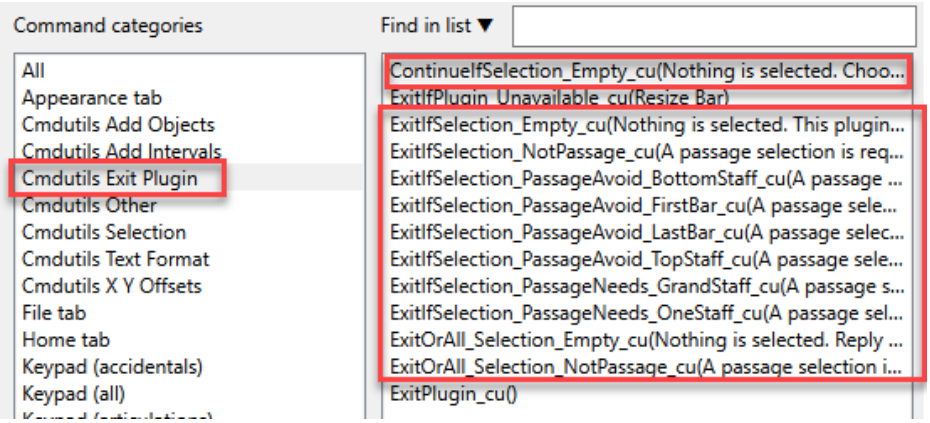
Select_First_NoteChordNoRest_cu()
Select_First_NoteChordNoRest_EachStaff_cu()
Select_First_NoteChordRest_cu()
Select_First_NoteChordRest_EachStaff_cu()
Select_First_Object_cu()
Select_First_Object_EachStaff_cu()
Select_First_Object_SystemOK_cu()

Select_Last_NoteChordNoRest_cu()
Select_Last_NoteChordNoRest_EachStaff_cu()
Select_Last_NoteChordRest_cu()
Select_Last_NoteChordRest_EachStaff_cu()
Select_Last_Object_cu()
Select_Last_Object_EachStaff_cu()
Select_Last_Object_SystemOK_cu()

ShiftSelectionNextBar_cu()
ShiftSelectionPreviousBar_cu()

TraceSelection_cu()

*Some of the **Exit Plugin** commands in **cmdutils** can warn you if there is a problem with the selection*



Cmdutils Selection Commands

Saving and Restoring the Selection

Possibly the most useful of the **cmdutils Selection** commands are commands that will save the current selection, or restore the previous selection.

SaveSelection_cu ()
- Saves the current selection so it can be restored later. This is most useful for a passage selection.

RestoreSelection_cu ()
- Restores the most recent selection saved by SaveSelection_cu.

These are mostly useful for passage selections, and you should be careful to be sure the saved selection is still meaningful before restoring it.

A lot of times plugins start with a passage selection, do a lot of processing, and then select individual objects that they have manipulated, and you might want to follow that plugin with another plugin that needs the original selection.

You cannot restore a selection if you saved one in a different score, and if you save a selection, then add or remove a lot of objects in the selection, and then try to restore the selection, you may not get a useful result. You cannot restore a selection without having saved one previously.

Here is a macro that adds a minor second interval to all selected notes, then filters the notes that were added, then **cuts** the added notes, so they are on the clipboard, but no longer in the score. The purpose of this macro is to put the new notes in the clipboard so they can be pasted to a different staff. Here is the basic code to do that.

```
AddInterval_Down_Minor_cu(2)
filter_bottom_note_for_deletion
cut
```

The first instruction adds a note at the interval of a minor second to each selected note. If the original note was part of a chord, the new note will be added below the lowest note in the chord, so the new notes will always be the lowest notes in the selection.

We filter the lowest notes, thus filtering the notes that were just added, and them cut them, which copies them to the clipboard and deletes them.

Assume we are starting with a passage selection, though it would also work with a non-passage selection. If there is no selection at all when these commands are run, however, nothing will happen.

After the filter command, only the filtered notes are selected, and after the **cut,** nothing is selected. You will see the original notes but they are no longer selected.

As a stand-alone macro, this might be fine, but if you wanted to do more processing with the original notes, it would be useful if they were still selected, but there is really no way to reconstruct the original selection at the end of the macro.

What we can do is save the selection before we do anything by using **SaveSelection_cu()**. We will then keep track of any changes made to the selection by each macro command.

After running **AddInterval_Down_Minor_cu(2)**, we check to see if it changed the selection in the course of adding notes. It appears that it just expands the selection, but if you were not sure that would always happen, you could call **RestoreSelection_cu()** so you would be sure of what the selection was at that point.

I have commented out a call to **RestoreSelection_cu** (using //) in the example below to show that I considered adding one, but decided it was not needed. I will often add and comment out code in the initial code-writing process, then clean up unneeded commands later.

filter_bottom_note_for_deletion will change the selection to only include the filtered notes, and **cut** will remove the selection altogether.

Since I saved the original selection, I can restore it with **RestoreSelection_cu()**, and we can start on another macro or plugin, as if nothing had happened to the selection.
Here is the final code:

```
SaveSelection_cu()
AddInterval_Down_Minor_cu(2)
// RestoreSelection_cu()
filter_bottom_note_for_deletion
cut
RestoreSelection_cu()
```

Without the **cmdutils** instructions (or the separately available **SaveSelection** and **RestoreSelection** plugins), I would have to stop after running the macro, and manually restore the original selection if I wanted to continue. This kind of defeats the purpose of automating the workflow.

Validating the selection

I mentioned that the example macro would do nothing if there were no selection, and that it really made more sense if it started with a passage selection.

I can use one of the **ExitPlugin** methods to the macro to check that the initial selection will work for the macro. If I add this command to the front of the macro, and nothing is selected, the plugin will put up a warning with the supplied text, and then stop.

```
ExitIfSelection_Empty_cu(Nothing is selected. This plugin will now exit.)
```

If I add this command to the front of the macro, and there is no **passage** selection, the plugin will put up a warning with the supplied text, and then stop.

```
ExitIfSelection_NotPassage_cu(A passage selection is required. This plugin will now exit.)
```

I would probably prefer to have there be a passage selection, as that gives me for flexibility for later commands, so I might end up with this as the final macro:

```
ExitIfSelection_NotPassage_cu(A passage selection is required. This plugin will now exit.)
SaveSelection_cu()
AddInterval_Down_Minor_cu(2)
// RestoreSelection_cu()
filter_bottom_note_for_deletion
cut
RestoreSelection_cu()
```

If the macro is run when there **is** a passage selection, there will be no warning, as one would expect.

Commands for selecting and unselecting objects

There are some other selection-related **cmdutils** commands that can also be useful. You can turn a bunch of individually selected objects into a passage selection, or a passage selection into separate selected objects. You can select the entire score in a number of ways. You can delete objects without deleting the staff, and for debugging, you can trace the structure of the selection.

MakePassageSelection_cu ()
- Turns a non-passage selection into a passage selection that includes all the previously selected objects, or converts a system selection to a passage selection.

MakeSystemPassageSelection_cu()
- Turns a non-passage selection into a system selection that includes all the previously selected objects, or converts a passage selection to a system selection.

FilterAllSelected_cu ()
- Turns a passage selection into a group of separately selected objects, as a filter would do.

DeleteSelection_cu()
- Deletes all the selected objects, but does not delete the staff even if the entire staff is selected.

SelectAll_Passage_cu()
- Makes a non-system passage selection of the entire score.

SelectAll_Passage_System_cu()
- Makes a system passage selection of the entire score.

SelectAll_NonPassage_cu()
- Makes a non-system passage selection of the entire score excluding objects in the system staff.

SelectAll_NonPassage_System_cu()
- Makes a non-passage selection of the entire score.

TraceSelection_cu ()
- Write a description of the current selection to the Plugin Trace Window. Can be useful for debugging.

Changing the size of a passage selection

Included in the **cmdutils Selection** commands are some that mimic Sibelius selection commands, such as

ExtendSelection_Left_cu ()
- Move the left end of the selection one note/rest to the left

ExtendSelection_Right_cu ()
- Move the right end of the selection one note/rest to the right

ExtendSelection_Down_cu()
- Move the bottom staff in the selection down 1 staff. Converts non-passage selection to a passage before extending.

ExtendSelection_Up_cu()
- Move the top staff in the selection up 1 staff. Converts non-passage selection to a passage before extending.

These commands do the opposite of the **Extend** commands, reducing the size of a passage selection

ContractSelection_Left_cu ()
- Move the right end of the selection one note/rest to the left

ContractSelection_Right_cu ()
- Move the left end of the selection one note/rest to the right

ContractSelection_Down_cu()
- Move the top staff in the selection down 1 staff. Converts non-passage selection to a passage before contracting.

ContractSelection_Up_cu()
- Move the bottom staff in the selection up 1 staff. Converts non-passage selection to a passage before contracting.

These commands will extend passage selections so they select entire pages, or fully select the first and/or last bar in the selection.

ExtendSelection_Pages_cu (score, selection)

- Get the first bar number on the page containing the first selected object, and the last bar number on the page containing the last selected object, and make a passage selection between those bar .

ExtendSelection_FullBar_Left_cu()
- Make the first bar in the selection be fully selected

ExtendSelection_FullBar_LeftRight_cu()
- Make the first and last bar in the selection be fully selected

ExtendSelection_FullBar_Right_cu()
- Make the last bar in the selection be fully selected

These commands select the first or last object or Note/Rest out of an existing selection. If the original selection was not a passage selection, these routines sort the objects so they will be in "score order" first (sorted by staff, then bar, then position in bar, then voice), and then the first or last of these is selected.

Select_First_NoteChordNoRest_cu ()
Select_First_NoteChordNoRest_EachStaff_cu ()
Select_First_NoteChordRest_cu ()
Select_First_NoteChordRest_EachStaff_cu ()
Select_First_Object_cu ()
Select_First_Object_EachStaff_cu ()
- These put the selected objects into chronological score order if needed, then select the first object in the selection or in each staff in the selection.

Select_Last_NoteChordNoRest_cu ()
Select_Last_NoteChordNoRest_EachStaff_cu ()
Select_Last_NoteChordRest_cu ()
Select_Last_NoteChordRest_EachStaff_cu ()
Select_Last_Object_cu ()
Select_Last_Object_EachStaff_cu ()
- These put the selected objects into chronological score order if needed, then select the last object in the selection or in each staff in the selection.

Commands that Exit a plugin or macro based on the selection state

These commands check whether there is no selection at all, or whether it is a passage selection or not, and exit the plugin or macro if the selection is not what you want it to be. In some cases you can tell it to continue anyway, or to select the entire score, and then continue.

ContinueIfSelection_Empty_cu (strMsgYesNoContinue)
- Exits the plugin if there is no selection and the user responds No in the message box. strMsgYesNoContinue is the message the user will see.

ExitIfSelection_Empty_cu(strmessage)
- Exits the plugin if there is no selection

ExitIfSelection_NotPassage_cu (strMessageIfNotPassage)
- Exits the plugin if there is no passage selection

ExitOrAll_Selection_Empty_cu(strMessageIfEmpty)
- Exits the plugin if there is no selection, or selects the entire score (non-system selection) and continues

ExitOrAll_Selection_NotPassage_cu(strMessageIfNotPassage)
- Exits the plugin if there is no passage selection, or selects the entire score (non-system selection) and continues

ExitIfSelection_PassageAvoid_BottomStaff_cu(strMessage)
ExitIfSelection_PassageAvoid_FirstBar_cu(strMessage)
ExitIfSelection_PassageAvoid_LastBar_cu(strMessage)
ExitIfSelection_PassageAvoid_TopStaff_cu(strMessage)
- These will exit the plugin or macro if there is a passage selection that includes a "forbidden" staff or bar.

ExitIfSelection_PassageNeeds_GrandStaff_cu(strMessage)
- This will exit the plugin or macro if there is a passage selection that does not contain the 2 staves of a grand staff instrument.

ExitIfSelection_PassageNeeds_OneStaff_cu(strMessage)
- This will exit the plugin or macro if there is a passage selection that contains anything other than a single staff.

Go To Commands

GoToFirstObject_cu()
- Selects the first staff object in the score (not a passage selection), and brings the selection into view.

GoToFirstObject_SystemOK_cu()
- Selects the first staff or system object in the score (not a passage selection), and brings the selection into view.

GoToLastObject_cu()
- Selects the last staff object in the score (not a passage selection), and brings the selection into view.

GoToLastObject_SystemOK_cu()
- Selects the last staff or system object in the score (not a passage selection), and brings the selection into view.

GoToFirstBar_cu()
GoToLastBar_cu()
GoToNextBar_cu()
GoToPreviousBar_cu()
- Makes a system passage selection in the desired bar, and brings the selection into view.

GoToNextPage_cu()
GoToPreviousPage_cu()
- Makes a system passage selection in first bar of the desired page, and brings the selection into view.