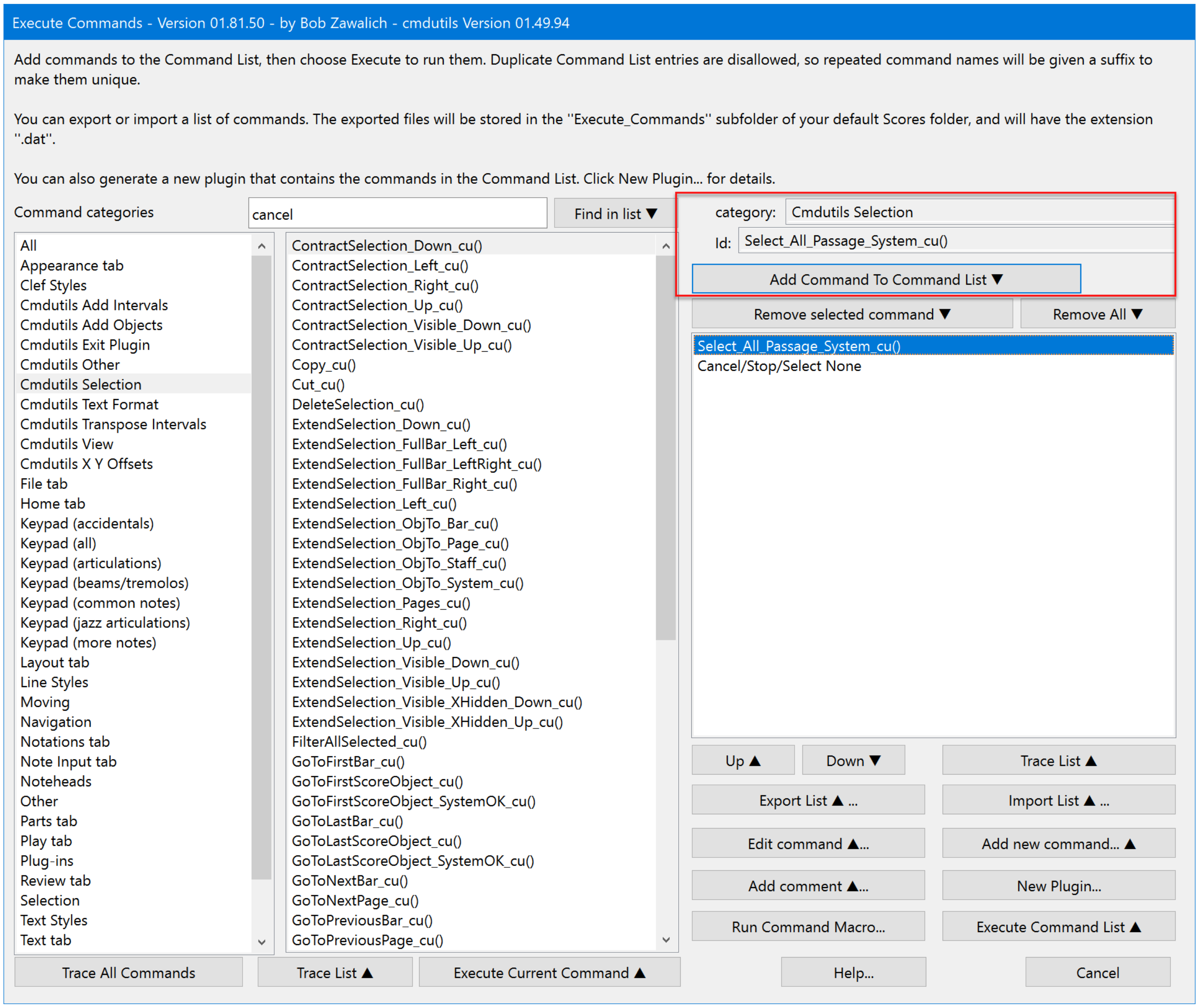
Execute Commands - Updates - Dialog Buttons and Setting the Focus

Bob Zawalich May 28, 2024

In the plugin **Execute Commands** one can define a sequence of commands to be run one after another. To do that you pick commands – which can be Sibelius commands, plugins, or cmdutils commands (commands contained in the separate plugin library **cmdutils.plg**) – and add them to the **Command List**, which is the rightmost list box in the plugin.

There are 3 big listboxes in the dialog, which will be called by these names:

- The leftmost list is **Command categories**
- The middle list is **Commands in Category**. It is the list of commands in the currently selected category in **Command Categories**.
- The leftmost list is the **Command List**, this list of commands that will be executed in sequence.



Add commands to the Command List, then choose Execute to run them. Duplicate Command List entries are disallowed, so repeated command names will be given a suffix to make them unique.

You can export or import a list of commands. The exported files will be stored in the "Execute_Commands" subfolder of your default Scores folder, and will have the extension ".dat".

You can also generate a new plugin that contains the commands in the Command List. Click New Plugin... for details.

cancel

category: Cmdutils Selection

Id: Select_All_Passage_System_cu()

Command categories:

All
Appearance tab
Clef Styles
Cmdutils Add Intervals
Cmdutils Add Objects
Cmdutils Exit Plugin
Cmdutils Other
Cmdutils Selection
Cmdutils Text Format
Cmdutils Transpose Intervals
Cmdutils View
Cmdutils X Y Offsets
File tab
Home tab
Keypad (accidentals)
Keypad (all)
Keypad (articulations)
Keypad (beams/tremolos)
Keypad (common notes)
Keypad (jazz articulations)
Keypad (more notes)
Layout tab
Line Styles
Moving
Navigation
Notations tab
Note Input tab
Noteheads
Other
Parts tab
Play tab
Plug-ins
Review tab
Selection
Text Styles
Text tab

ContractSelection_Down_cu()
ContractSelection_Left_cu()
ContractSelection_Right_cu()
ContractSelection_Up_cu()
ContractSelection_Visible_Down_cu()
ContractSelection_Visible_Up_cu()
Copy_cu()
Cut_cu()
DeleteSelection_cu()
ExtendSelection_Down_cu()
ExtendSelection_FullBar_Left_cu()
ExtendSelection_FullBar_LeftRight_cu()
ExtendSelection_FullBar_Right_cu()
ExtendSelection_Left_cu()
ExtendSelection_ObjTo_Bar_cu()
ExtendSelection_ObjTo_Page_cu()
ExtendSelection_ObjTo_Staff_cu()
ExtendSelection_ObjTo_System_cu()
ExtendSelection_Pages_cu()
ExtendSelection_Right_cu()
ExtendSelection_Up_cu()
ExtendSelection_Visible_Down_cu()
ExtendSelection_Visible_Up_cu()
ExtendSelection_Visible_XHidden_Down_cu()
ExtendSelection_Visible_XHidden_Up_cu()
FilterAllSelected_cu()
GoToFirstBar_cu()
GoToFirstScoreObject_cu()
GoToFirstScoreObject_SystemOK_cu()
GoToLastBar_cu()
GoToLastScoreObject_cu()
GoToLastScoreObject_SystemOK_cu()
GoToNextBar_cu()
GoToNextPage_cu()
GoToPreviousBar_cu()
GoToPreviousPage_cu()

Select_All_Passage_System_cu()
Cancel/Stop/Select None

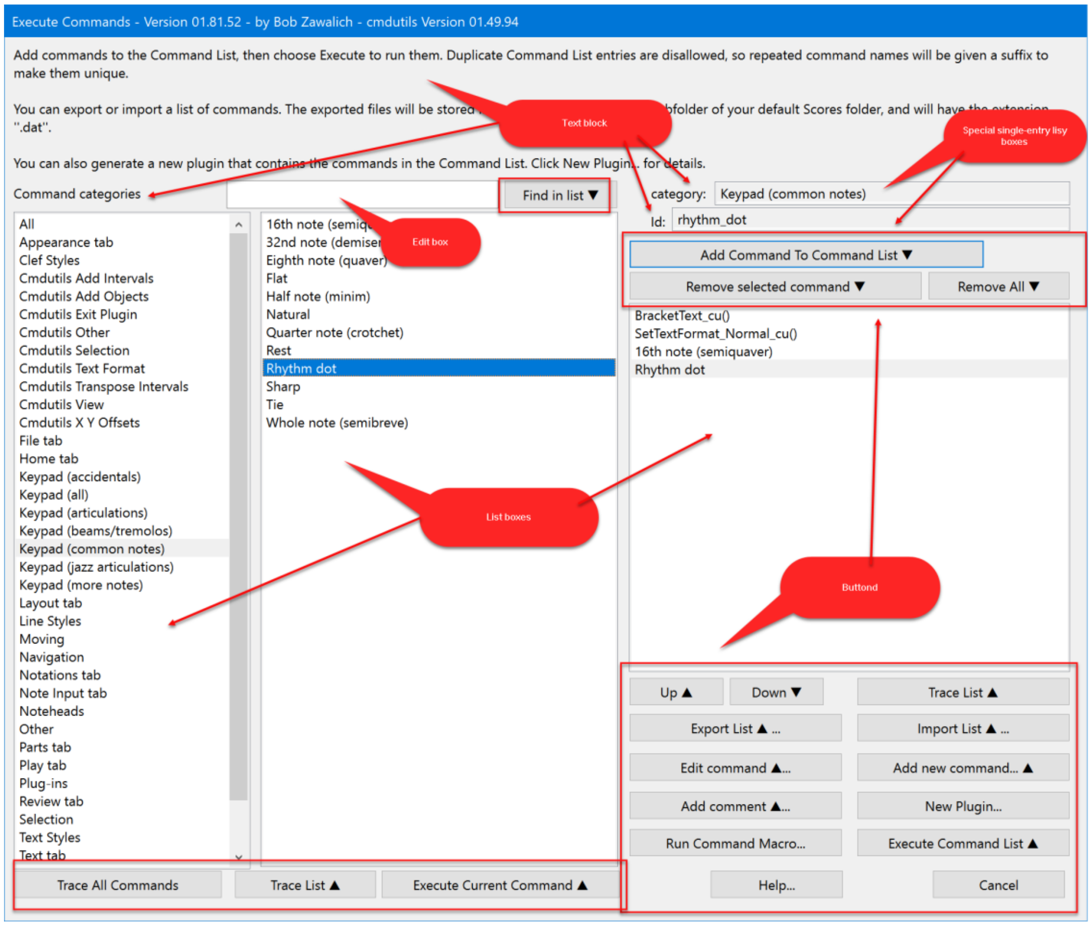
How Execute Commands works

In the simplest terms, you choose a command category in **Command categories**, and that would cause **Commands in Category** to fill with the commands contained in that category. You then select a command by scrolling down the alphabetized list, or using the **Find in list** button, and once selected, you would press **Enter** or click on the button **Add Command to Command List**. The chosen command will appear at the bottom of the **Command List**, and you can use the **Up** and **Down** buttons to move the command selected in the **Command List** up or down compared to the other entries in the list.

Once all the commands are in the **Command List** you can test the sequence by pressing the **Execute Command List** button. You can save the sequence by using **Export List**, and the resulting file can be brought back using **Import List**. The file created in **Export List** can be run by the plugin **Run Command Macro**. You can also use the **New Plugin…** button to generate a ManuScript plugin that will run the commands in sequence.

Dialog buttons and which control gets the focus

A plugin dialog can have a number of controls, including Text blocks, list boxes, edit boxes, radio buttons, and checkboxes. **Execute Command**'s dialog has everything except radio buttons and checkboxes, and the dialog controls are annotated in this screenshot.



Execute Commands - Version 01.81.52 - by Bob Zawalich - cmdutils Version 01.49.94

Add commands to the Command List, then choose Execute to run them. Duplicate Command List entries are disallowed, so repeated command names will be given a suffix to make them unique.

You can export or import a list of commands. The exported files will be stored bfolder of your default Scores folder, and will have the extension ".dat".

You can also generate a new plugin that contains the commands in the Command List. Click New Plugin... for details.

Command categories

Find in list ▼

category: Keypad (common notes)

Id: rhythm_dot

Add Command To Command List ▼

Remove selected command ▼ Remove All ▼

All
Appearance tab
Clef Styles
Cmdutils Add Intervals
Cmdutils Add Objects
Cmdutils Exit Plugin
Cmdutils Other
Cmdutils Selection
Cmdutils Text Format
Cmdutils Transpose Intervals
Cmdutils View
Cmdutils X Y Offsets
File tab
Home tab
Keypad (accidentals)
Keypad (all)
Keypad (articulations)
Keypad (beams/tremolos)
Keypad (common notes)
Keypad (jazz articulations)
Keypad (more notes)
Layout tab
Line Styles
Moving
Navigation
Notations tab
Note Input tab
Noteheads
Other
Parts tab
Play tab
Plug-ins
Review tab
Selection
Text Styles
Text tab

16th note (semiq
32nd note (demise
Eighth note (quaver)
Flat
Half note (minim)
Natural
Quarter note (crotchet)
Rest
Rhythm dot
Sharp
Tie
Whole note (semibreve)

BracketText_cu()
SetTextFormat_Normal_cu()
16th note (semiquaver)
Rhythm dot

Up ▲ Down ▼ Trace List ▲
Export List ▲ ... Import List ▲ ...
Edit command ▲... Add new command... ▲
Add comment ▲... New Plugin...
Run Command Macro... Execute Command List ▲
Help... Cancel

Trace All Commands Trace List ▲ Execute Current Command ▲

Any control except for Text blocks can get the focus. A control may be given the initial focus when the dialog comes up, and it can get the focus when you click on a control or Tab to it. When a control has the focus its appearance changes. The selected item in a listbox might change color, as in the **Rhythm dot** entry in the screenshot. Other controls might be marked with a different border to indicate having the focus.

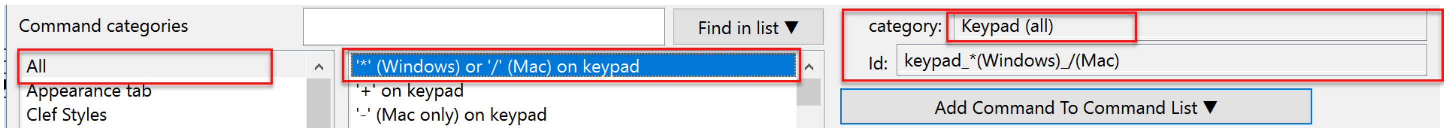
If any **button** has the focus it can be activated by pressing the **Enter** key. Most dialogs have a default button.

If no other button has the focus, pressing **Enter** will activate the default button. In the screenshot above, the default button, **Add Command to Command List**, is marked with a blue border (in Windows). If you press **Enter** in that dialog, **Add Command to Command List** will be activated, and would add the **Rhythm dot** command to the **Command List**.

Clicking on a button moves the focus to that button. If no button has the focus and you press Enter, the default button is activated, but the focus does not move to the button.

Initial focus handling in Execute Commands

The first time the plugin comes up in a Sibelius session, the first entries in **Command categories** and **Commands in Category** are selected, and **Commands in Category** has the focus. The default button is (as it always is), **Add Command to Command List.**



category: Keypad (all)
Id: keypad_*(Windows)_/(Mac)

The **category:** and **id:** controls above the **Add Command to Command List** button display the category and id of the selected command in **Commands in Category**. The category will usually match the selection in **Command categories**, but if **All** is selected there, the **category:** control will show the actual category for the selected command, which in the case is **Keypad(all)**.

Control activation and the 3 list boxes

In most cases setting the focus to one of the 3 listboxes will call a "callback routine" in the listbox control. This routine will change the settings in the **control:** and **id:** controls and will change the command that will be added to the **Command List**.

The exception to this rule in that if you click on the ***scroll bar*** *of one of the listboxes, the list box will gain the focus, and its selected item will turn blue (on Windows), but the callback routine is not called, and the command to be added is not changed.*

When I refer to these listboxes in this document I will use the term **activate** to indicate that the control's callback routine has been called, rather than saying that the listbox has the focus.

I will refer to the command that will be added to **Command List** as the **active command**.

If you want to **activate** a list box that has a scroll bar, you should click on the current list box entry instead of the scroll bar. You can also activate a list box by using the **Tab** key to move the focus to that control.

If you activate the **Command categories** list, its callback routine will change the contents of the **Commands in Category** list, which will select the first entry in **Commands in Category** and activate the list, even though it will not get the focus. This will change the **active command**. The **control:** and **id:** controls will be updated, but the focus will remain in the **Command categories** listbox.

Command categories will not be activated if you only click in the scroll bar.

Special Handling for the **Add Command to Command List** button

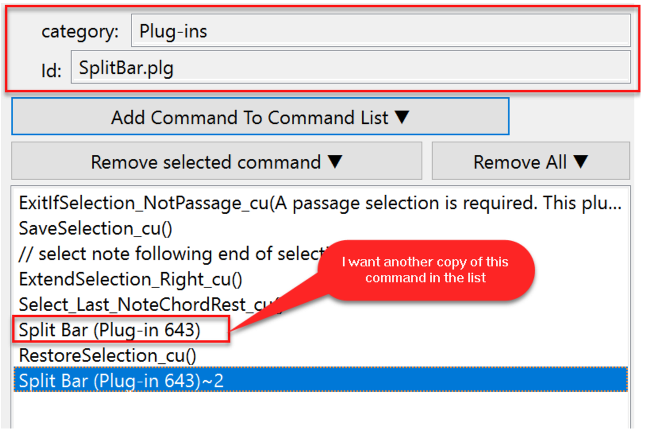
Add Command to Command List will make a copy of the active command, which could be the selected command in either **Commands in Category** or **Command List.** It will add that command to the bottom of the **Command List**.

*The last of these list boxes to be activated determines which command is the **active command**.*

Cloning a Command List entry using Add Command to Command List

Previously, the **active command** was always the most recently selected command in **Commands in Category**. As of May 2024, the **active command** can be a command selected in the **Command List**. If the active command is in **Command List**, adding a command will create a clone of the selected command in the **Command List**.

How to Tell which command will be added to Command List


category: Plug-ins
Id: SplitBar.plg

Add Command To Command List ▼
Remove selected command ▼
Remove All ▼
I want another copy of this command in the list

ExitIfSelection_NotPassage_cu(A passage selection is required. This plu...
SaveSelection_cu()
// select note following end of select
ExtendSelection_Right_cu()
Select_Last_NoteChordRest_cu(
Split Bar (Plug-in 643)
RestoreSelection_cu()
Split Bar (Plug-in 643)~2

The 2 controls immediately above the **Add Command To Command List** button display the category and language-independent **command id** of the **active command.**

The **active command** will be updated when either of the list boxes **Commands in Category** or **Command List** is activated**.**

Add Command to Command List will add the active command to the **Command List**.

Buttons and focus handling in Execute Commands

If you click on a button, rather than using **Enter** to activate it, the focus will normally move to the clicked button.

Execute Commands prefers to keep the focus in the middle listbox (**Commands in Category**) or the rightmost listbox (**Command List**). If you click on a button, **Execute Commands** performs the action associated with the button, and then moves the focus to an appropriate listbox.

This is how the different buttons handle the focus:

These buttons close the dialog, so the focus is irrelevant

- **Cancel**
- **Run Command Macro…**
- **Execute Command List**
- **Execute Current Command**

These buttons always move the focus to the **Command List**, regardless of where the focus had been

- **Up**
- **Down**
- **Trace List (in Command List block)**
- **Export List…**
- **Import List…**
- **Edit Command…**
- **Add New Command…**
- **Add Comment…**
- **New Plugin…**

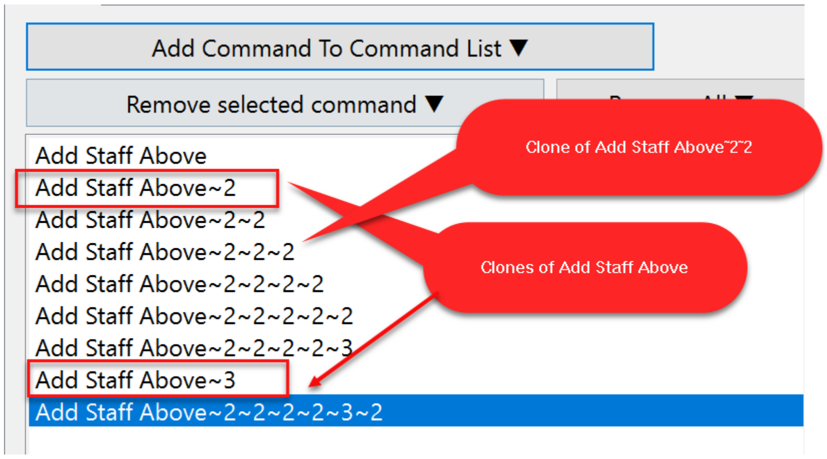
These buttons always move the focus to **Commands in Category**

- **Trace All Commands**
- **Trace List**

These buttons have special handling when they are activated

- **Add Command to Command List**
- **Help…**
    - These 2 buttons return the focus to the listbox that was most recently activated. For example, **Command in Category** was the most recently activated listbox, even if it had not had the focus, **Command in Category** will get the focus.
- **Remove selected command**
    - This returns focus to the **Command list**, unless **Command List** is empty; if empty it moves the focus to **Commands in Category**.
- **Remove all**
    - **Command List** will be empty so it moves the focus to **Commands in Category**.
- **Find in list**
    - This is quite different from all the other buttons.
    - This button keeps the focus so you can easily find the next match by pressing the **Enter** key
    - If you want to add a found command to the **Command List**, press **Tab** to put the focus in **Commands in Category**, then press **Enter** to activate the default button, or click **Add Command to Command List.**

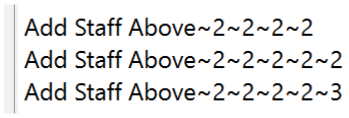
What are those “~2” things in the Command List?


Add Command To Command List ▼
Remove selected command ▼
Clone of Add Staff Above˜2˜2
Clones of Add Staff Above

Add Staff Above
Add Staff Above~2
Add Staff Above~2~2
Add Staff Above~2~2~2
Add Staff Above~2~2~2~2
Add Staff Above~2~2~2~2~2
Add Staff Above~2~2~2~2~3
Add Staff Above~3
Add Staff Above~2~2~2~2~3~2

Entries in the **Command List** must have unique text, so if you add a command that is already in the list, its name will get a suffix starting with ~2. If **Execute Commands** sees that that changed name already exists, it changes the suffix to ~3, ~4… until the name is unique. In the example above, **Add Staff Above~2** and **Add StaffAbove~3** are clones of **Add Staff Above**. After **Add Staff Above~2** had been created and I cloned **Add Staff Above** again, the plugin tried the ~2 suffix, which was not unique so it upped the number to 3.

If I clone a command that already has a suffix, the new suffixes are appended to the existing suffixes. Here, the clones start from **Add Staff Above~2~2~2~2** and add new suffixes.



Add Staff Above~2~2~2~2
Add Staff Above~2~2~2~2~2
Add Staff Above~2~2~2~2~3

The suffixes are only needed while the commands are in the list box; the suffixes are stripped when the commands are actually used. If you delete commands, the existing suffix numbers are not updated, so the suffixes may seem out-of-order, but the order does not matter. The only important thing is that there must be no duplicate entries in the list box.