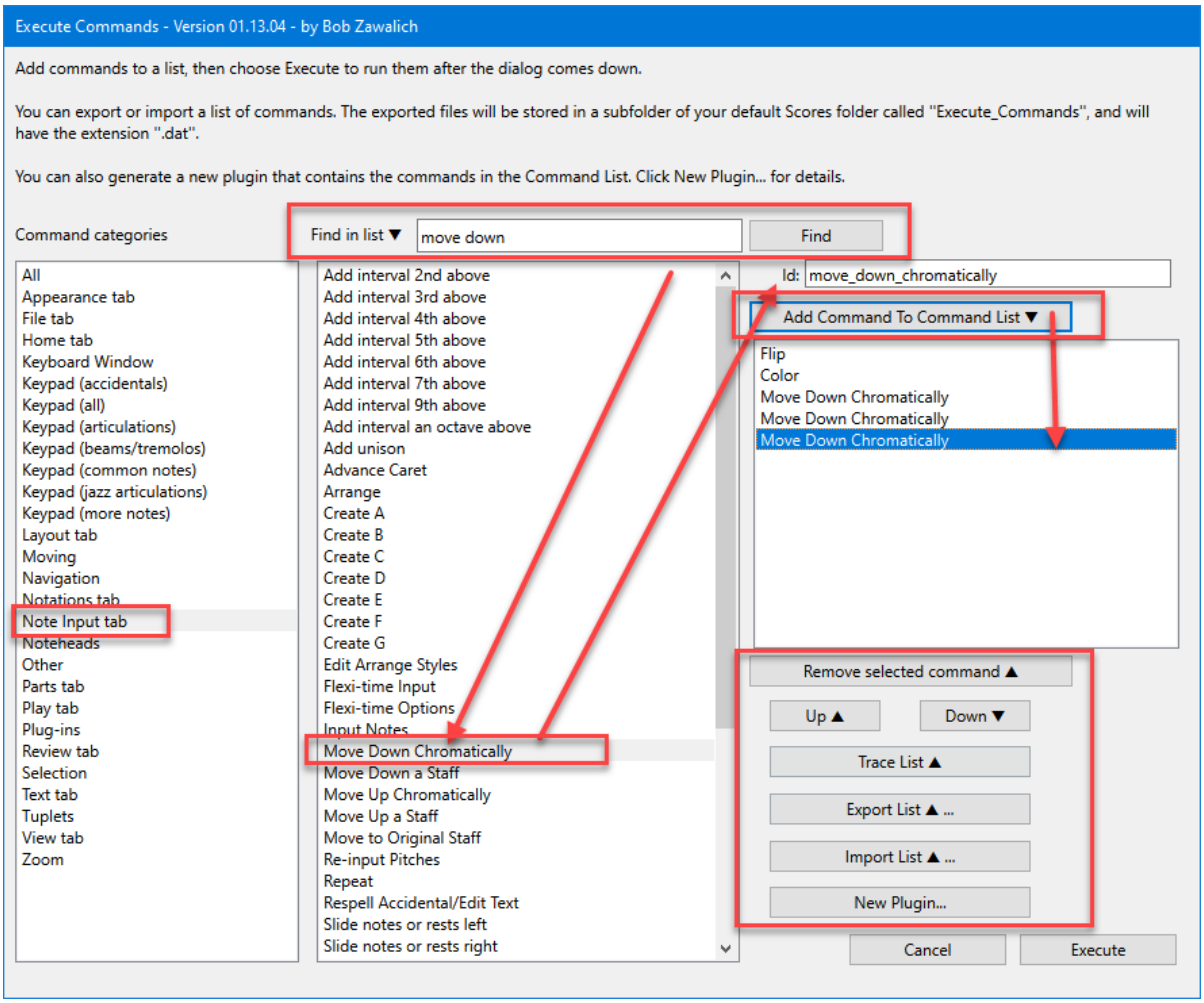
Plugins that work with Commands in Sibelius Ultimate 2021.2 and Later

Bob Zawalich 27 February 2021

The Execute Commands Plugin



The goal of the Execute Commands plugin is to make it easier to create "Command macros" or simple plugins that use the commands added in Sibelius Ultimate version 2021.2, without having to directly write code.

When you run the plugin you will see the dialog above, There are 3 listboxes.
The first 2 are very similar to the listboxes in File>Preferences>Keyboard Shortcuts.

1. The Command Categories list
2. The Commands in Category list

If you change the category in the first list box the Commands in Category list will be filled with the commands from that category.

When a command or category changes, the **"Id:" edit box** will be filled with the language-independent "Command Id" that corresponds to the selected command name. Some commands, notably the Plug-ins category will have no available Command Id. The Command Id is placed in an edit box to make it easier to copy it for use in other plugins.

The 3rd and rightmost listbox is the **Command List**. You can add the command that is currently selected in the **Commands in Category** list to the bottom of the **Command List** by activating the **Add Command to**

Command List button. This is almost always the default button, so you can usually just type **Enter** to add a command.

You can delete the selected entry in the Command List using **Remove Selected Command**. Move the selected command up or down in the Command List with the **Up** and **Down** buttons.

These controls should let you add and rearrange the commands in the **Command List**.

Once you have the commands the way you want them in the **Command List**, you can

- **Execute** the full set of commands with the **Execute** button. This will take down the dialog after running the commands.
- **Export** the list of commands to a text file (with .dat extension) with **Export List...**", which will be stored in the "Execute_Commands" subfolder which will be added to your default **Scores** folder.
- **Import** any of the text files you have exported **Import List...**", which will replace the contents of the Command List
- **Trace** the contents of the list with **Trace List**. This will trace the commands 3 ways:
    - The commands as shown in the list box
    - Plugin instructions for those commands, using language-dependent command names.
    - Plugin instructions for those commands, using language-independent command ids, where possible.

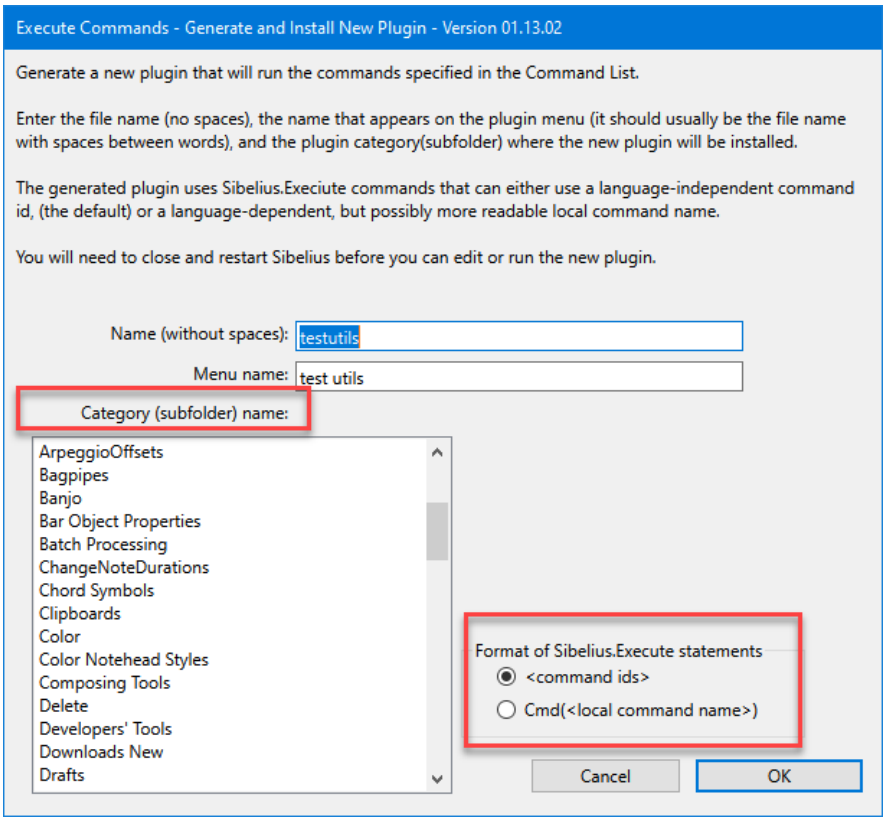
    - Here are the Traced lines for the list shown in the dialog above. Commands that are plugins have a different form, but we will discuss that later.

```
Flip
Color
Move Down Chromatically
Move Down Chromatically
Move Down Chromatically

Sibelius.Execute(Cmd("Flip"));
Sibelius.Execute(Cmd("Color"));
Sibelius.Execute(Cmd("Move Down Chromatically"));
Sibelius.Execute(Cmd("Move Down Chromatically"));
Sibelius.Execute(Cmd("Move Down Chromatically"));

Sibelius.Execute("flip"); // Flip
Sibelius.Execute("color"); // Color
Sibelius.Execute("move_down_chromatically"); // Move Down Chromatically
Sibelius.Execute("move_down_chromatically"); // Move Down Chromatically
Sibelius.Execute("move_down_chromatically"); // Move Down Chromatically
```

- **Generate a plugin file** using **NewPlugin...**
    - This will take the commands in the Command List and write out a new plugin file that will execute these commands. This will save you from having to get the ManuScript syntax correct among other things.
    - The plugin file will be added to any of the plugin subfolders you have on your machine. The plugin will create an **Execute_Commands** subfolder in your user **Plugins** folder as a convenient place to keep such plugins.
    - You will need to close and restart Sibelius in order to run the new plugin or edit it in the Sibelius plugin editor.



The New Plugin dialog in Execute Commands

Try the **Find box** in Execute Commands. It is really cool. You can type in any part of a command name, and it will try to find it in the All Commands list. If it finds a match, it looks up the category for the command and switches over the list boxes to be in that category. This way you can both find all the similar command names and see their category as well.

Language issues

If you are running commands from the Ribbon, or add several commands to the **Command List** in **Execute Commands** and then choose **Execute**, you can use the command names as they are shown, translated into the current language.

If you export and import macros, or create new plugins, then the command names will only work if you are running on a machine in the same language. This is probably how it will be most of the time.

However, if you have a set of useful macros files, or plugins that used Commands, and want to share them with someone running Sibelius in another language, the command names will not be recognized.

"Move Down Chromatically" will not be found on a German machine.

To deal with this problem, command names are also given language -independent **Command Ids**.
If your plugin uses **Command Ids,** it will run in any language.

The problem is that to use them you have to know the command id that corresponds to a specific local command name. **Trace List** in **Execute Commands** will map these for you automatically when possible, so in the example about you see:

```
Sibelius.Execute("move_down_chromatically"); // Move Down Chromatically
```

"move_down_chromatically" in the **Command Id** for "Move Down Chromatically", or its equivalent in any other language.

If you need to find the mapping, Sibelius provides a list. You can also use the plugin **Execute Command Ids To Names Plugin**, which will trace out a set of comma separated fields showing the **Command Id, Command Name**, and **Category** for all commands that have **Command Ids**. It is intended to be loaded into a spreadsheet and sorted as desired.

Commands that are plugins do not have command ids.

If, for example, you want to write a macro that runs several plugins in a row, you cannot call **Sibelius.Execute(Cmd("Plugin menu name"));**

because the Cmd() function will return an empty string for plugin command names.

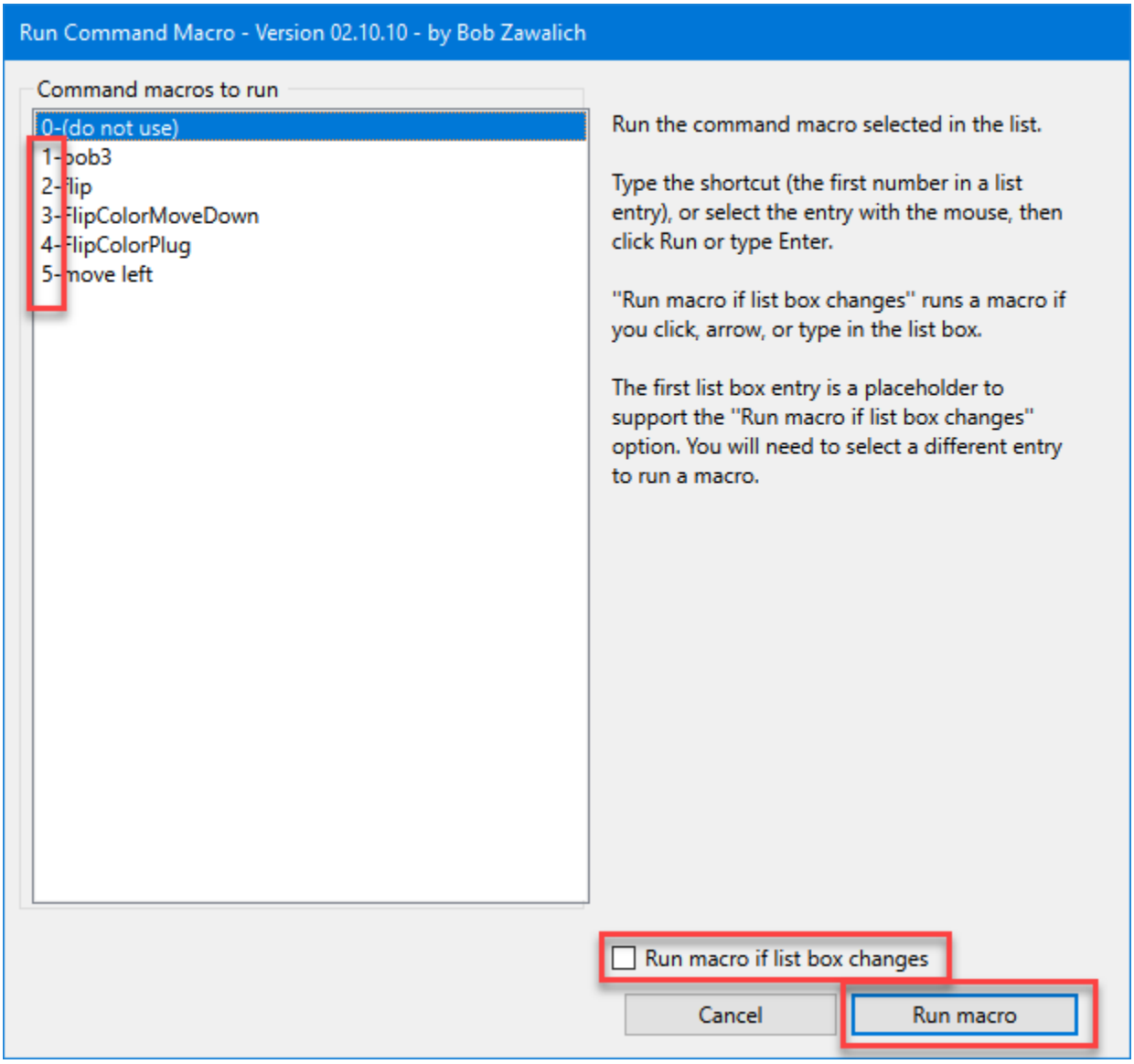
The way I am getting around this is that I added a routine called **RunPluginFromCommandName** to the shipping **utils.plg** plugin

In **Execute Commands** when I need to output an instruction for a plugin I generate a call to **utils.RunPluginFromCommandName,** such as

```
utils.RunPluginFromCommandName('Add Capo Chord Symbols (Plug-in 818)');
```

The disadvantage of this method is that it only knows the local command name (since there are no **Command Ids** for plugin commands), and saved plugins or macros that use calls to shipping plugins will only run in the language in which they were written. But they do at least run in the original language, and calls to non-shipping plugins, which generally do not have their menu names translated, should work in any language,

The Run Command Macro plugin



This plugin provides a more convenient way to run macros you created by using **Export List...** in **Execute Commands**.

The list will contain any data files that you exported. Each has a number shortcut in the list.

You can select an entry in the list and then click on **Run Macro**, and the macro will be executed.

If you check the **Run macro if list box changes** checkbox, then typing a numeric shortcut or clicking on a list entry or arrowing up or down in the list will immediately run that macro. This is a little tricky to get to work as you want, but it is very fast when it is done correctly.

Entry "0" in the list is a placeholder. You can't run it, and it is only there to work around problems with the implementation of list boxes when **Run macro if list box changes** is checked**.**

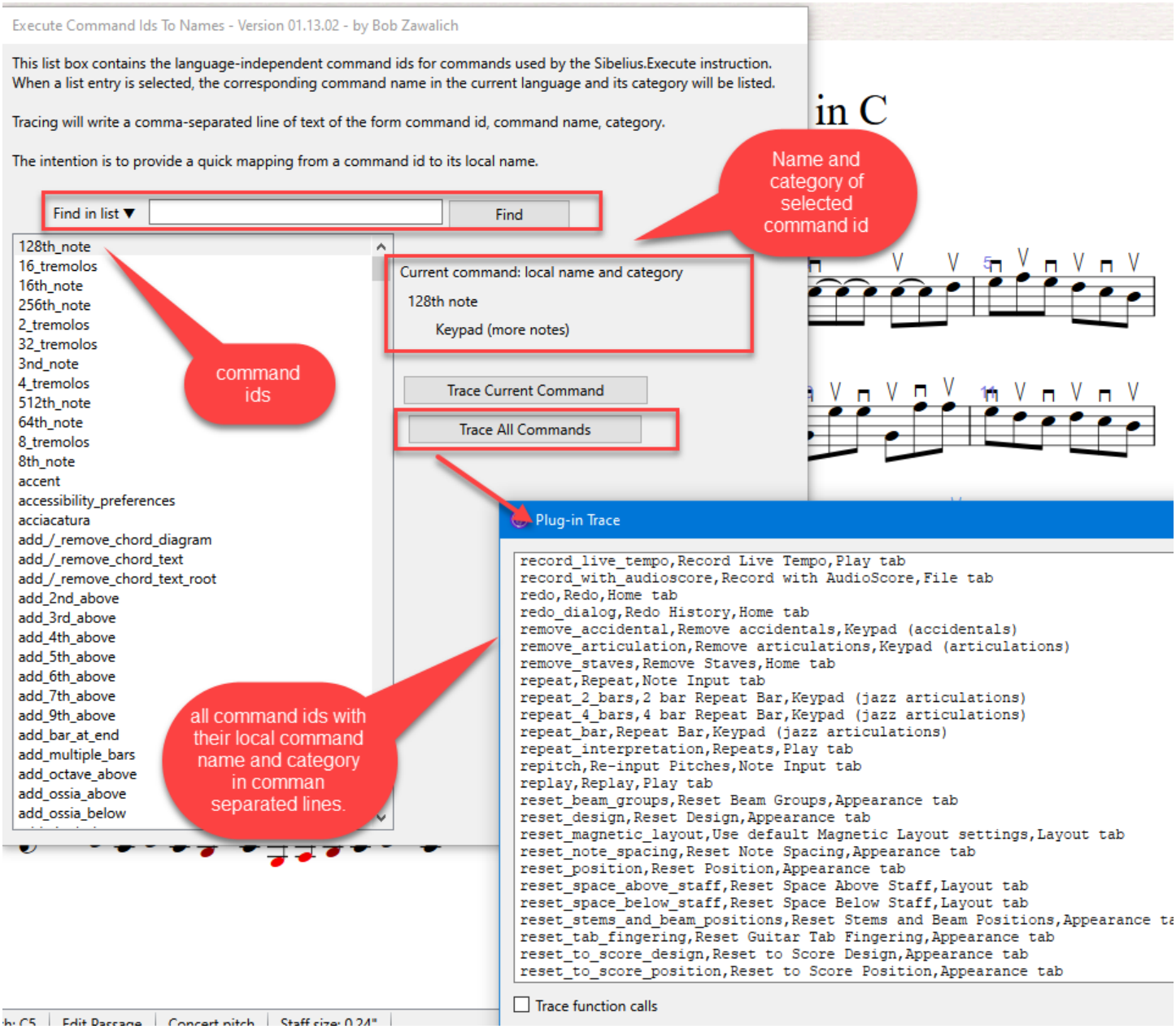
The Execute Command Ids To Names Plugin

If you use **Trace List** or **New Plugin** in **Execute Commands**, it will try to translate the local command names into **Command Ids**. I think this is the easiest way to map a command name to a Command Id.

However, there is a document that gives the command names and command ids, and there is also the plugin **Execute Command Ids To Names**.

It gives you a list of all the supported command ids. Selecting a list entry will display its name and category, and if you use Trace All Commands, it will write out the command id, command name, and category for each command id.

My intention is that you could copy this into a spreadsheet, and then sort the columns as you wish, so you can get the list ordered several different ways.



The Trace Current Command button will trace the command id, command name, and category for each of the currently selected command id, and it will also trace Sibelius.Execute instructions for the command, for copying into a plugin you are writing.

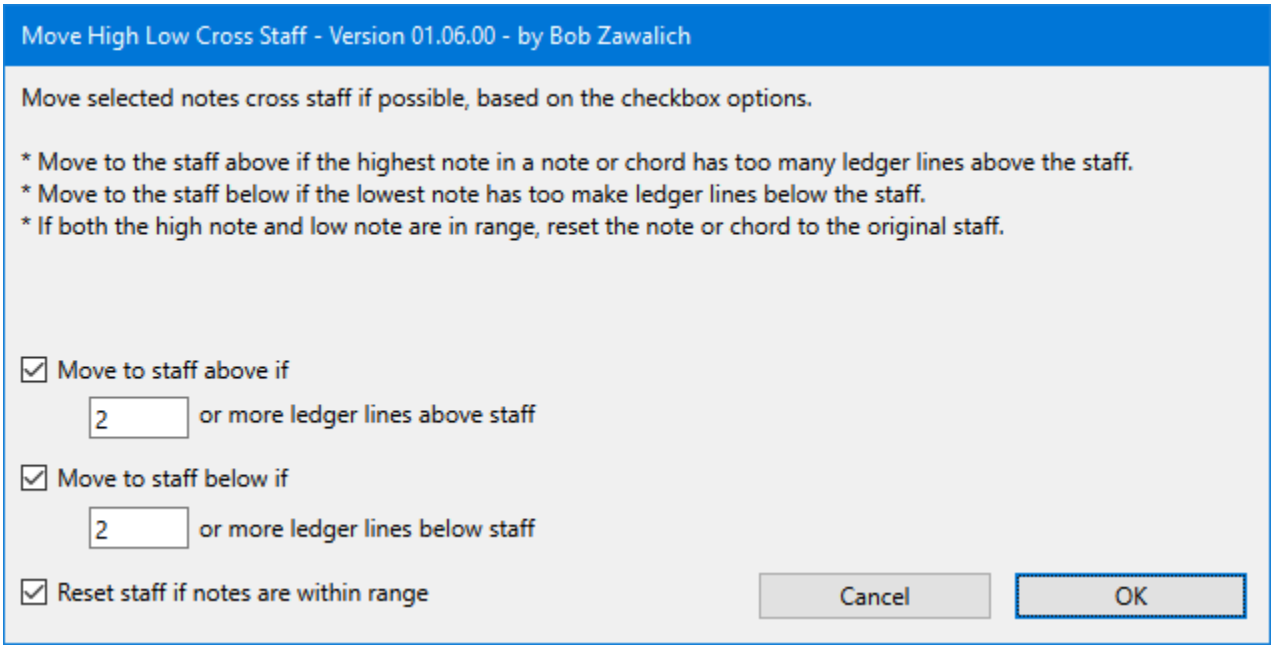
Here is an example of the output for Trace Current Command for the "engraving_rules" Command ID.

Note that the command that uses the Command Name displays the Command Id in a comment, and the one that uses the Command Id shows the Command Name in a comment.

```
engraving_rules,Engraving Rules,Appearance tab
Sibelius.Execute("engraving_rules"); // Engraving Rules
Sibelius.Execute(Cmd("Engraving Rules")); // engraving_rules
```

A Plugin that incorporates Commands, but is not simply sequences of commands

- MoveHighLowCrossStaff



  - This will move selected notes that are in staves that are appropriate sources for cross staff beaming to the staff above if the highest note in the note or chord has too many ledger lines, or moves to the lower staff is there are too many ledger lines below the staff.
  - It uses the commands
    - cross_stave_move_up
    - cross_stave_move_down
    - cross_stave_reset