Tutorial: Execute Commands and cmdutils

Adding Text, Lines, and Symbols to a score

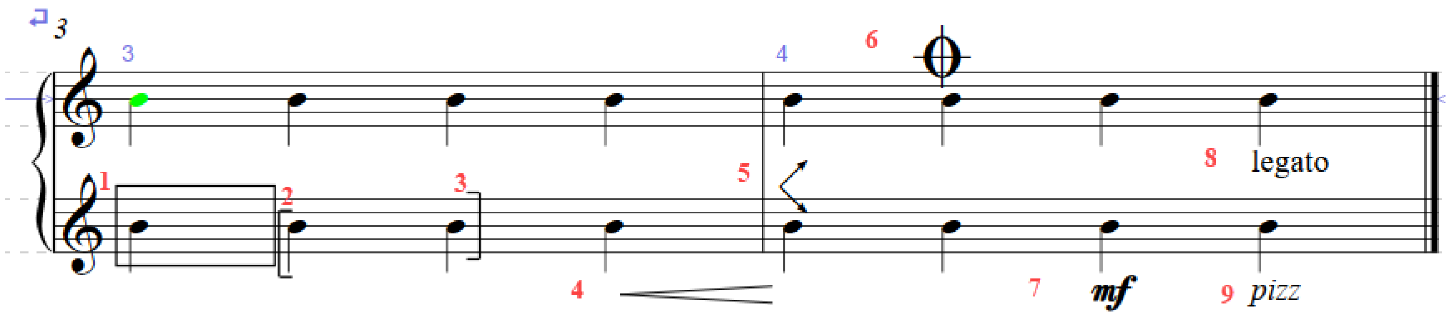
Bob Zawalich September 22, 2021

Here are some examples of commands that will add objects at the current selection. Originally the first note in the lower staff was selected, and Select next object commands were executed after each **Add** command.

1. **Add_Line_Box_cu()**
2. **Add_Line_Bracket_Vertical_Left_cu()**
3. **Add_Line_Bracket_Vertical_Right_cu()**
4. **Add_Line_Hairpin_Crescendo_cu()**

5. **Add_StaffSymbol_cu(Choral divide arrow)**
6. **Add_SystemSymbol_cu(Coda)**

7. **Add_Text_Dynamics_cu(mf)**
8. **Add_Text_Expression_cu(pizz)**
9. **Add_Text_Technique_cu(legato)**



Add Object and "Add and Select" Methods

These add objects to the current selection, mimicking the way Commands work. These routines use the current selection of **Sibelius.ActiveScore**, but they all call lower-level routines where the score, selection, and certain more advanced features may be explicitly specified. I have tried to make the top level routines use as few parameters as possible.

Add Object methods typically return the object created.

If the AddSelect form is used, the created object will be selected (non-passage selection).

If the add fails the selection will be cleared. You can call **ContinueIfSelection_Empty_cu(strMsgYesNoContinue)**, or **ExitIfSelection_Empty_cu(strMessageIfEmpty)** to exit the plugin if the selection had been cleared, indicating that the add had been unsuccessful.

Using the **AddSelect** forms in combination with **SaveSelection_cu** and **RestoreSelection_cu** can be an effective way to add and manipulate an object, and then continue, and this will be described in detail below.

If you are using these commands in your own plugin, you need to be careful about adding objects into a selection while you are walking through the selection in a plugin's "for each" loop, since adding or deleting objects will change the selection and could disturb the loop.

There are several **Line** and **Text** style commands that include the style of the object in the command name.

For Lines, you can also call **Add_Line_cu(styleTextOrId),** which adds a line of the requested line style. **styleTexOrId** must be a StyleAsText or StyleId string which is valid in the score, spelled EXACTLY as ManuScript expects it to be. It is very easy to get the **styleTextOrId** wrong, so only use this if you really need the flexibility.

The commands **Add_Text_Dynamics_cu(strText), Add_Text_Expression_cu(strText),** and **Add_Text_Technique_cu(strText)** include the text style in the command. If you need more Text styles than these , you will need to run 2 commands.

Run T**extStyleDefaultForCommands_cu(styleTextOrId)** first to set the text style to be used, then call **Add_Text_cu(strText)** to actually add the text.

If you do not explicitly set the text style, **Add_Text_cu** will use **Technique** text style. The text style set by **TextStyleDefaultForCommands_cu** will remain active for the remainder of the Sibelius session unless it is changed by another call to **TextStyleDefaultForCommands_cu.** I recommend calling **TextStyleDefaultForCommands_cu** before each call to **Add_Text_cu.**

Text formatting wildcards such as \B\ and \I\ can be included in **strText**. Wildcard text produced by **New Plugin** will have doubled backslashes (\\B\\ and \\I\\ in this case), but macros keep the single backslashes.

Identifying objects to be added (style names and style ids)

You can apply **Notehead Styles** or add symbols in **Execute Commands**, but you need to know the **Style** or other identifier for the object you are creating, and it must be spelled EXACTLY as ManuScript expects it to be.

In general, objects have a language-specific name (often called **StyleAsText**) and a language-independent identifier, often called a **StyleId**. To make macros or plugins as portable as possible, you should use the **StyleId** rather than the **StyleAsText**, but either will work in the same language in which the macro or plugin was written. User-defined styles also will not work if that style is not defined in the current score.

If you know the **StyleAsText** name for Lines, Text, Symbols, or Noteheads, you can find the **StyleId** or index for such objects by running one of these Trace commands, editing the command you wish to run to contain the name, and copying and pasting from the trace window.

Trace_LineStyleIdFromName_cu(Glissando (wavy))
Trace_NoteStyleIndexFromName_cu(Diamond)
Trace_SymbolIndexFromName_cu(Mordent)
Trace_TextStyleIdFromName_cu(Technique)

- These can be used with a properly spelled name to get the language-independent StyleId of Index for the name.

Now you can edit the Add command to use the language-independent id or index.

Alternatively, you can select some objects (Lines, Text, Notes, Symbols) in the score that you want the identifier for and call:

Trace_Object_Type_Name_StyleOrIndex_cu()

- This can be used by selecting appropriate objects in your score to get the language-independent StyleId of Index for the objects. This is probably the easiest way to find such ids.

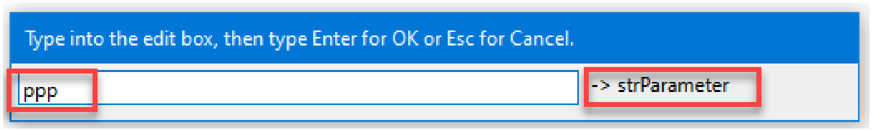
Acquiring the text or style name from the user

The **Add** commands assume that you know the value of the parameter to the commands, so you might say **Add_Text_Dynamics_cu(**mf**).** If you would like the user to be able to choose the text, you can use the command **GetUserInput_cu(strVariable)** to put up a dialog with an edit box the user can type into.

GetUserInput_cu uses a clever technique I call a *parameter variable* that lets it pass its input to another command. ***Its parameter is the name of a variable that can be used as the parameter to any other command that has a parameter***. As a simple example, I could write:

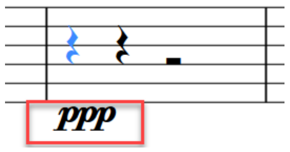
```
GetUserInput_cu(strParameter)
Add_Text_Dynamics_cu(strParameter)
```

When I run these commands I would see this dialog box, and would type the text I wanted (**ppp** in this example), then press **Enter**.


Type into the edit box, then type Enter for OK or Esc for Cancel.

ppp

-> strParameter

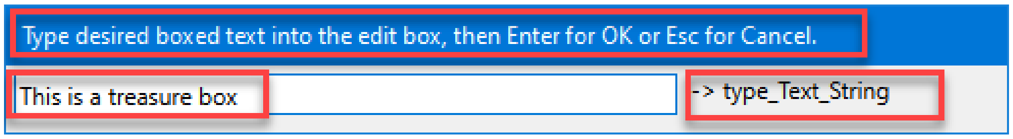
I would then see this in the score at the start of the current selection: 

GetUserInput_cu stores what I typed (**ppp**) into a special variable that I named **strParameter**, and **Add_Text_Dynamics_cu** uses that variable instead of a specific piece of text. When you choose the name for a parameter variable be sure to pick something that will not be confused with “normal” text.

For a somewhat more complex example, we can set up the **GetInput_cu** dialog to have a more specific heading line, and can use **TextStyleDefaultForCommands_cu** to choose a text style that is not available in the special **AddText** commands:

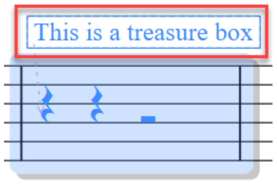
```
SetUserInput_Heading_cu(Type desired boxed text into the edit box, then Enter for OK or Esc for Cancel.)
GetUserInput_cu(type_Text_String)
TextStyleDefaultForCommands_cu(text.staff.boxed)
Add_Text_cu(type_Text_String)
```

The dialog comes up with a custom header, and the name of the parameter variable is displayed. I can type in some text:


Type desired boxed text into the edit box, then Enter for OK or Esc for Cancel.

This is a treasure box

-> type_Text_String

And I will see: 

If I wanted to I could make a second **GetUserInput_cu** call to ask for the text style, but since it has to be typed exactly correctly, I avoid that when I can.

Add Object Methods

Add_Line_cu(styleTextOrId)

- Adds a line of the requested style. styleTexOrId must be a StyleAsText or StyleId valid in the score, spelled EXACTLY as ManuScript expects it to be.

Add_Line_8va_cu()
Add_Line_Box_cu()

Add_Line_Bracket_Vertical_Left_cu()
Add_Line_Bracket_Vertical_Right_cu()
Add_Line_Ending_First_cu()
Add_Line_Ending_Second_cu()
Add_Line_Hairpin_Crescendo_cu()
Add_Line_Hairpin_Diminuendo_cu()
Add_Line_Plain_cu()
Add_Line_Slur_cu()
Add_Line_Trill_cu()
Add_Line_Vertical_cu()

- Adds a line of the specified style to the selection
- Some of the lines, especial vertical lines like brackets and box lines are given slightly different (and better, in my opinion) positions that Sibelius uses in the Lines menu.

Add_Line_Full (score, selection, styleTextOrId, fSelectNewObject)

- Adds a line of the requested style. styleTexOrId must be a StyleAsText or StyleId valid in the score, spelled EXACTLY as ManuScript expects it to be.

Add_StaffSymbol_cu (nameOrIndexSymbol)
Add_SystemSymbol_cu (nameOrIndexSymbol)
Add_Symbol_Full (score, selection, nameOrIndexSymbol, fUseSystemStaff, fSelectNewObject)

- Adds a SymbolItem or SystemSymbolItem to the selection. nameOrIndexSymbol is either a symbol name or an index into the symbol table, spelled EXACTLY as ManuScript expects it to be.

Add_Text_cu(strText)

- Adds text using the text style that was set by running the **TextStyleDefaultForCommands_cu(styleTextOrId)** command, or adds Technique text if the text style is not set.
- It is best to always run **TextStyleDefaultForCommands_cu** immediately **before** running this command.

Add_Text_Dynamics_cu (strText)

- Add a dynamics text object with Expression text style and **MusicText** character style, so the text will be correctly formatted.

Add_Text_Expression_cu (strText)

- Add an Expression text object to the selection

Add_Text_Technique_cu (strText)

- Add a Technique text object to the selection

Add_Text_Full (score, selection, strText, styleTextOrId, fSelectNewObject)

- Add a text or system text object (determined by the style id). styleTexOrId must be a StyleAsText or StyleId valid in the score

ApplyNoteheadStyle_cu (strIdNote)
ApplyNoteheadStyle_Full (score, selection, strIdNote)

- Changes the notehead style of all selected notes. strIdNote is either a numeric index or a valid Note Style Name, spelled EXACTLY as ManuScript expects it to be.

TextStyleForAddText_cu(styleTextOrId)

- Stores the value of strtTextOrId for the remainder of the Sibelius session so it can be used by **Add_Text_cu** and **AddSelect_Text_cu**
- It is best to always run this immediately **before** running **Add_Text_cu** or **AddSelect_Text_cu**

Trace_Object_Type_Name_StyleOrIndex_cu()

- This can be used by selecting appropriate objects in your score to get the language-independent StyleId of Index for the objects. This is probably the easiest way to find such ids.

Trace_LineStyleIdFromName_cu(Glissando (wavy))
Trace_NoteStyleIndexFromName_cu(Diamond)
Trace_SymbolIndexFromName_cu(Mordent)
Trace_TextStyleIdFromName_cu(Technique)

- These can be used with a properly spelled name to get the language-independent StyleId of Index for the name.

“Add and Select” Methods

These work the same way as the **Add** methods except that at the end the added object will be the only thing selected. It can be useful to call **SaveSelection_cu** before making such a call, and then after having manipulated the new object you can restore the previous selection using **RestoreSelection_cu.**

For example, you can create a Text object with **Add_Text_Technique_cu (strText),** but you may want to change some properties that were not set when you created the object. If you wanted the text to be bold, you could write something like this:

SaveSelection_cu()
AddSelect_Text_Technique_cu(I am a bold one!)
SetTextFormat_Bold_cu()
RestoreSelection_cu()

which adds and selects a **Technique** Text object, makes it bold, and then restores the original selection.

AddSelect _Line_cu(styleTextOrId)

AddSelect _Line_8va_cu()
AddSelect _Line_Box_cu()
AddSelect_Line_Bracket_Vertical_Left_cu()
AddSelect_Line_Bracket_Vertical_Right_cu()
AddSelect_Line_Ending_First_cu()
AddSelect_Line_Ending_Second_cu()
AddSelect_Line_Hairpin_Crescendo_cu()
AddSelect_Line_Hairpin_Diminuendo_cu()
AddSelect_Line_Plain_cu()
AddSelect_Line_Slur_cu()
AddSelect_Line_Trill_cu()
AddSelect_Line_Vertical_cu()

AddSelect_StaffSymbol_cu (nameOrIndexSymbol)
AddSelect_SystemSymbol_cu (nameOrIndexSymbol)

AddSelect_Text_cu(strText)

- Adds text using the text style that was set by running the **TextStyleDefaultForCommands_cu(styleTextOrId)** command, or adds Technique text if the text style is not set.
- It is best to always run **TextStyleDefaultForCommands_cu** immediately **before** running this command.

AddSelect_Text_Dynamics_cu (strText)
AddSelect_Text_Expression_cu (strText)
AddSelect_Text_Technique_cu (strText)

Some example macros

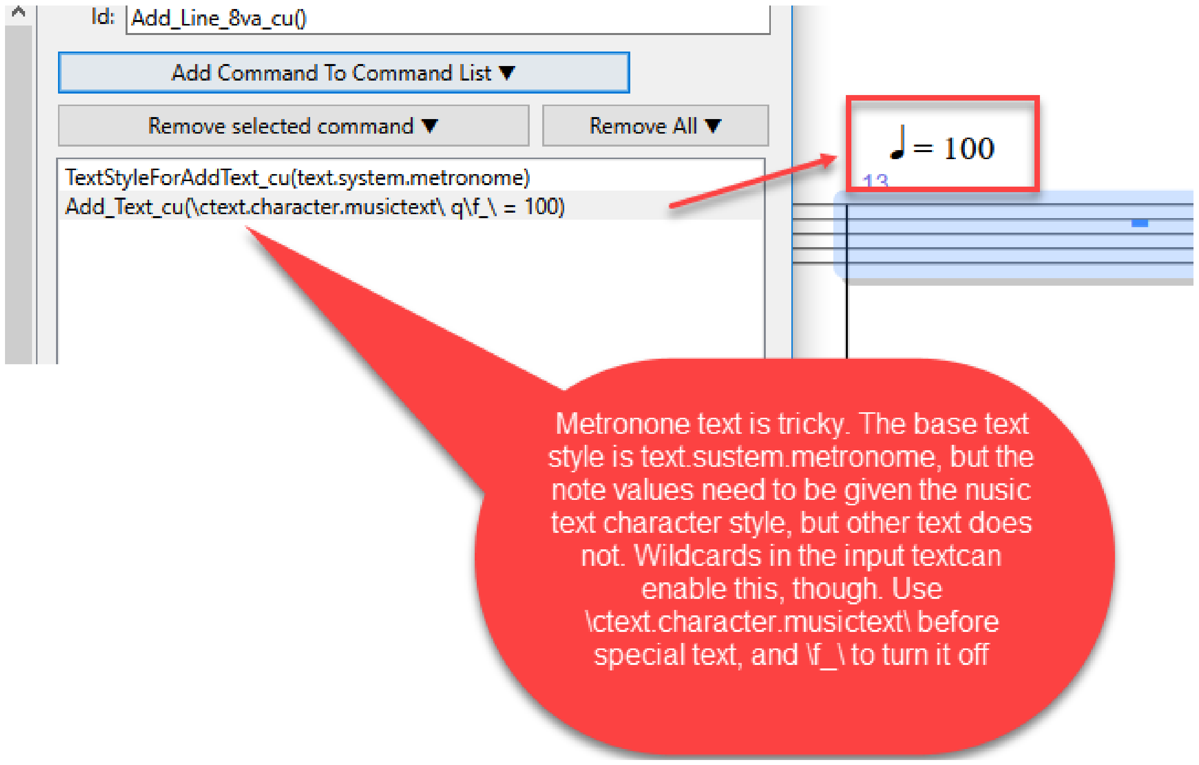
With some cleverness, you can expand what is possible to create. For example, you might think of creating some Metronome text by running

```
TextStyleDefaultForCommands_cu(text.system.metronome)
Add_Text_cu(q = 100)
```

but this will print a “q” rather than a quarternote symbol. Real metronome text makes a font change (or character style change) to the q.

It turns out that text formatting wildcards will work in the case. You can use **\ctext.character.musictext**\ to change the q to the desired Music Text character style, and **\f_**\ to turn it off. So this works fine:

```
TextStyleDefaultForCommands_cu(text.system.metronome)
Add_Text_cu(\ctext.character.musictext\q\f_\ = 100)
```



```
TextStyleForAddText_cu(text.system.metronome)
Add_Text_cu(\ctext.character.musictext\ q\f_\ = 100)
```

Metronone text is tricky. The base text style is text.sustem.metronome, but the note values need to be given the nusic text character style, but other text does not. Wildcards in the input textcan enable this, though. Use \ctext.character.musictext\ before special text, and \f_\ to turn it off

Here are 2 similar macros that add text using wildcards

1. Add Footer File Date
   - The add a piece of text with the Text Style **Footer (inside edge)** to the first bar in the score
   - The Text Style uses the language-independent StyleId rather than the English StyleAsText
   - The text is these wildcards: **\n\\$FILEPATH\ \$FILEDATE**
   - I added a linefeed to the start to position it further from the end of the score
   - The date is the date the file is printed.

```
// Add Footer File Date
// by Bob Zawalich
//
GoToFirstBar_cu()
TextStyleDefaultForCommands_cu(text.system.page_aligned.footer.inside)
AddSelect_Text_cu(\n\\$FILEPATH\  \$FILEDATE\)
goto_selection_start
```

2. Add Wildcards Title Subtitle Composer Lyricist
    - Bora asked me if this would be possible and it seemed that it could be done in a similar way to Add Footer File Date.
    - It can be useful if you have a score without the Title/Subtitle/Composer/Lyricist fields.
    - It adds 4 pieces of system text using the Title, Subtitle, Composer, and Lyricist text styles
    - It then brings up a messages box to warn you of what will happen next
    - It brings up the Score Info part of the File Tab so you can fill in Text for these fields
    - You have been warned to click on the Home tab of the Ribbon when you are done.
    - The Text Styles use the language-independent StyleIds rather than the English StyleAsTexts
    - It will not remove any existing text for these fields
    - If you do not fill in the Score Info for a field, the text object will be present but invisible.

```
// Add Wildcards Title Subtitle Composer Lyricist
// by Bob Zawalich -  concept by Ilkay Bora Oder
// Add 4 text objects using wildcards then bring up Score Info to fill in the text
//
// This will make a system selection in the first bar of the score
GoToFirstBar_cu()
TextStyleDefaultForCommands_cu(text.system.page_aligned.title)
Add_Text_cu(\$Title\)
TextStyleDefaultForCommands_cu(text.system.page_aligned.subtitle)
Add_Text_cu(\$Subtitle\)
TextStyleDefaultForCommands_cu(text.system.page_aligned.composer)
Add_Text_cu(\$Composer\)
TextStyleDefaultForCommands_cu(text.system.page_aligned.lyricist)
Add_Text_cu(\$Lyricist\)
goto_selection_start
MessageBox_cu(Please fill in the Score Info fields for the title, subtitle, composer, and lyricist, then click the Ribbon Home
tab.)
score_info
```