Using ConsoleTools in Dorico

If, as is likely, you are considering to use the ConsoleTools framework as a way of accessing Dorico's Lua functionality in, first of all, an easy-to-use fashion, the amount of documentation below may be somewhat daunting. Don't be discouraged, though – ConsoleTools has been designed to make simple things simple. At the same time, the framework aims to be „super-chargeable" when extending it with the full power of the Lua language, should you feel so inclined.

You will hopefully find that, for the vast majority of your interactions with the framework, relying on the keycommands alone – as set up in the keycommands JSON file (see above) – is sufficient once you have set up a tool collection that suits your needs. Setting up a collection is detailed in the section Tool Setup File, further below.

The initial tool setup (tutorialToolSetup.lua) is intended as a practical introduction, allowing you to familiarise yourself with the basic principles of how to use the framework. The tutorial is designed within the constraints of the free-to-download version of ConsoleTools.

Tutorial: keycommands and basic console use

The following section is a walk-through of the basic commands for interacting with the framework. It assumes the tutorial tool setup, as well as the specific keycommands shown on the previous page. Do **not** try to run the tutorial file or any of the included files directly (via the **Script** menu) – with proper installation, the framework will initialise automatically upon first use of one of its keycommands.

For your very first steps with ConsoleTools after installing it, open a Dorico project with some notation (or simply enter some notes into a new project). Note: while all edits within the tutorial are trivial, it is recommended to make a backup if you use an existing file for trying things out.

In Write Mode, open the properties panel (at the bottom of the window) and select a note. Then use the keycommand K for tool slot 1.

The first thing that you will notice is that Dorico's Script Console opens up and displays a number of messages (if this does not happen, you probably already ran a script within the current Dorico session; in that case, bring up the console manually). You can safely ignore the console output for now. Nonetheless, it is recommended to leave the Script Console open when working with ConsoleTools, and to arrange it somewhere on the screen where it is visible while working in the Dorico project window.

NB: The first time the Script Console is brought up, it will cause the main application's focus to switch, and you must return focus to the project window manually before any further keycommands will register again. While there is currently no way to prevent this from happening, it also will occur only once during a Dorico session.

Next, check the properties panel – you should find that the color for the selection has been changed to red. Initially, slot 1 holds several tools for setting the Color property, and you have just applied one of them. There is a keycommand available for listing a slot's current tools; Ctrl+F1, Ctrl+K (that is: while keeping Ctrl pressed down, press F1 and K) will print this to the Script Console's output field:

```
Slot 1 currently contains:
  >>            ! to red
                ! to blue
                ! to transparent
```

The currently active tool is marked with >>. An exclamation mark preceding a tool's name denotes that this tool will be executed immediately upon becoming the slot's current tool.

Try out the keykommands for moving between the three tools of slot 1: Ctrl+Alt+K and Ctrl+K will cycle forward and backward, and Ctrl+Alt+Shift+K will switch to the previously used tool.

Use the keycommand L for slot 2. It holds a single tool, for deactivating the color property.

You can set up tool profiles in ConsoleTools, i.e.: presets of tool-to-slot allocations, tailor-made for certain tasks. The tutorial setup contains three such tool profiles. Use the keycommand `Ctrl+Alt+,` to cycle to the next one. Since no profile is active at first, this will result in the first profile being selected for application, with this message printed to the Script Console:

```
Profile 'harmonics' (#1) to be applied | (2 tool slots)
```

With a note selected, use `K` again. Slot 1 now holds a different tool, which sets the Harmonics Type property for the note to “Artificial”. The tool can do more, though. Press `K` twice in quick succession, and the Partial property is set to 4. Doing it three, four, five times sets the property to 3, 5 and 2, respectively, placing the most common artificial harmonics conveniently at your fingertips. Tool slot 2 (keycommand `L`) also holds a new tool now, for turning the Harmonics Type property off again.

Use the keycommand `Ctrl+Alt+,` once more, to cycle to the next tool profile:

```
Profile 'offsets / velocity' (#2) to be applied | (2 tool slots)
```

Using the keycommands for the two tool slots, you will notice that nothing happens. That is, nothing except the following message being printed to the Script Console:

```
! no tempo set
```

This profile binds together a number of different tools for adjusting playback-related properties. It also goes already slightly beyond the mere sending of static command strings to Dorico, of the kind that you may be accustomed with if you have experimented with Dorico’s macro recording capability. Instead, the property values are calculated in relation to a tempo, which must be provided first. To do this, switch to the Script Console and, in the lower field, enter:

```
bpm(80)
```

Go back to the project window and, with a note selected, use `K` once more. This time, the two properties Playback start offset and Velocity in the Notes and Rests panel have been turned on and set to 640 and 10, respectively. Return to the Script Console and enter another tempo value:

```
bpm(100)
```

Back in the project window, use `K` again; the Playback start offset property has now been set to 800, a different value than before.

To find out more about a tool, you can display its documentation, if any was provided. This can be done via the console (documented further below), but there is also a keycommand available for each slot to display documentation for its current tool. `Ctrl+F2, Ctrl+K` prints the following:

```
->Help for current tool of slot 1
  Description: apply portamento
     - sets the 'Notes and Rests'>'Playback start offset' property of a selected note, calculated as …
     - sets the 'Notes and Rests'>'Velocity' property of a selected note to 10
```

For a reminder of which tools are currently applied, you can use the keycommand `Ctrl+F1, Ctrl+F3`, which at this point will print:

```
* Contents of all current slots:
Slot 1 currently contains:
>>              apply portamento
Slot 2 currently contains:
>>              apply legato | slow
                apply legato | fast
```

Using the keycommand L does now also result in properties being changed. If you cycle to the second slot's next tool (Ctrl+Alt+L), you will notice that there is a message printed to the console:

```
Current tool for slot 2: apply legato | fast
```

… but there has been no actual change to the playback properties. That is because, unlike with the color tools earlier, the two tools in the second slot are not set to be executed immediately (thus, in the list above, they are not marked with an exclamation mark). For such a tool, you must use the slot's main keycommand after having changed the tool: pressing L again will now apply a different set of values for the Playback start offset and Velocity properties. Whether tools are executed immediately or not will depend very much on context and vary between different profiles.

Tool profiles are convenient, but there is a more flexible way of applying tools to slots, if needed. Let's assume you want to apply both the "slow legato" and the "fast legato" playback adjustments from above to several passages in a score, and you also want to mark up each case by color. For such a case, you can use tools *ad hoc*, by typing something like this into the console input field:

```
set(1,true,"idLS","idG")
set(2,true,"idLF","idB")
```

(You can simply copy-and-paste these two lines. They represent one of several ways to allocate tools via the console, which are documented in depth further below.)

The keycommand Ctrl+F1, Ctrl+F3 can once again be used to see how the slot allocation has changed:

```
* Contents of all current slots:
Slot 1 currently contains:
>>          !  apply legato | slow
            !  to green
Slot 2 currently contains:
>>          !  apply legato | fast
            !  to blue
```

With this configuration, you can simply select a passage and then cycle twice through the relevant slot – reducing to two actions what would normally take a lot of clicking and typing. (If you try this particular example out in practice, make sure to filter for notes first, if needed, just as when doing these operations manually. Eventually, of course, you could also set up your own tool for filtering notes, from a Dorico command string, and incorporate it in the cycle.)

The keycommand Ctrl+F1, Ctrl+F1 prints all available tools to the Script Console, so you do not have to rely on your memory when making *ad hoc* allocations. Each tool is listed with a numerical index, and, if provided, a unique string index, both of which can be used to match it to a slot.

There are ways to use tools directly through the console. One of them is calling tools that are bound to Lua variables. For example, selecting a note and then entering:

```
tr()
```

… into the console will execute the "to transparent" tool from the beginning of the tutorial, even if that tool is not allocated to any slot at all. Likewise, the `bpm` example from before was a variable-bound tool, not intended to be used through a tool slot ever.

As said earlier, ConsoleTools aims to make simple things simple. Nonetheless, users (as well as script developers) can do more complex things with it. The following section about Console use functions documents these capabilities in more detail. In the concluding Tool Setup File section you can learn how to build and manage your own personalised collection of tools and profiles, with many of the examples of this tutorial revisited for that purpose.