A while ago, Ed Hirschman wrote to the tech chat page about a plugin he had written to hide the extension lines of trills.

He mentioned that it did not have the error checking one might want, but that it worked fine. His code looked like this:

```
**********************
for each obj in Sibelius.ActiveScore.Selection
{
if ((obj.IsALine) and (obj.StyleId = 'line.staff.trill'))
{
obj.Duration = 0;
}
**********************
```

Yeah, it would fail ungracefully if someone did something like open Sibelius with no scores and attempted to run it. But this is very unlikely.

Someone on the list complained that you had to select the trills for it to work, but it was described as getting rid of all of them. I suggested just changing the description to say it would get rid of selected trills, but the other person came back saying he thought that the plugin should offer a user friendly alert if nothing was selected.

I actually disagreed, and said that such requirements are an unnecessary bar to people publishing plugins. I then proceeded to prove my point by showing the tedious process that it takes to get a plugin to "production state".

But all in all it is not so tedious, and you can set it up so that you can reuse such code many times, which is what I do. So I though I would extract the explanation here, and offer it as a workable model for any plugin.

One of my personal goals is to write a plugin so it will never fail ungracefully. So I always add a couple of checks to the start of a plugin:

```
**************************************
if (Sibelius.ScoreCount = 0)
{
Sibelius.MessageBox(_ScoreError);
return False;
}

score = Sibelius.ActiveScore;
```

```
if (score.StaffCount = 0)
{
Sibelius.MessageBox(_ScoreError);
return False;
}
*****************************************
```

So now the plugin would look like:

```
*********************************
if (Sibelius.ScoreCount = 0)
{
Sibelius.MessageBox(_ScoreError);
return False;
}

score = Sibelius.ActiveScore;

if (score.StaffCount = 0)
{
Sibelius.MessageBox(_ScoreError);
return False;
}

for each obj in Sibelius.ActiveScore.Selection
{
if ((obj.IsALine) and (obj.StyleId = 'line.staff.trill'))
{
obj.Duration = 0;
}
*************************************
```

where _ScoreError is a bit of data, something like:

_ScoreError "Please open a score with one or more staves and run the plugin again."

So now we want to add code to handle when nothing is selected. There is no really easy way to do this. In the old days I would sometimes look to see if there was a passage selection, and if so I would process the selection. If not I would process the entire score. But then I realized that people often do a multi-selection, ctrl/cmnd clicking a few items. There is no way to actually see if that is happening without looking to see if anything is selected.

So you can write another method called IsEmptySelection that looks like this:

```
****************************************
IsEmptySelection "(score) {
for each obj in score.Selection
{
return False;
}

return True; // only if nothing in selection
****************************************
```

Depending on the situation, you might want to have it only check selected items that are not in the system staff. That might be a good choice in this case, because trills will never be in the system staff. So if they just selected the title, you might want to treat it as an empty selection. So you end up with

```
************************************
IsEmptySelection "(score) {
// only consider locations outside system staff

for each obj in score.Selection
{
if (obj.ParentBar.ParentStaff.IsSystemStaff = False)
{
return False;
}
}

return True; // only if nothing in selection
************************************
```

(Added for programmers only) There is one additional check I always put here to avoid the unlikely case where only a barline is selected. In that case, you get the barline back from such a loop, but it is not a bar object (I argue it should not be returned in such a loop, but it is). If you try to look at its type, the plugin will fail. So to avoid this, add an IsObject check:

```
************************************
IsEmptySelection "(score) {
// only consider locations outside system staff

for each obj in score.Selection
{
if (obj.ParentBar.ParentStaff.IsSystemStaff = False)
{
```

```
if (IsObject(obj)) // handle when only a barline is selected
{
return False;
}
}
}

return True;  // only if nothing in selection
}"
**************************************
```

OK, so that is figured out and you can tell if nothing is selected. Now you need to ask the user if they want it to apply to the entire selection or not. (You could just run on the entire score but that is sometimes a surprise). So let us be polite and ask:

```
**********************************
fProcessEntireScore = False;
if (IsEmptySelection(score) = True)
{
fContinue = Sibelius.YesNoMessageBox(_msgSelectWholeScore); //True means Yes, continue and process score
if (fContinue = False)
{
return False; // they said no, so stop the plugin
}
fProcessEntireScore = True;
}
**********************************
```

You will need to define

_msgSelectWholeScore "There is no selection; Please click Yes to process the entire score, or No to stop the plugin."

as well.

And now we know they want to process the entire score.

But how do you do that? If you are willing to process each staff separately and determine that objects are actually selected (the latter is easy since Sib 5 and very difficult before that), then if you know the trick, you can swap the Score object for the Selection object and rewrite your original loop to process staves and instead of going through the selection. You need to be careful not to use specific Selection object properties if you do that, though, or you will crash.

But since Sib 5, you can change the selection, and so you can save off the original selection, select the entire score, process everything and then restore the selection. So the main loop could look like this:

```
****************************************
score = Sibelius.ActiveScore;
selection = score. score.Selection;
selection.StoreCurrentSelection();
score.Redraw = False; // this saves a lot of time avoiding redrawing each time the score changes

if (fProcessEntireScore = True)
{
selection.SelectPassage(1, score.SystemStaff.BarCount, 1, score.StaffCount); // select all staff items
}

for each obj in selection
{
if ((obj.IsALine) and (obj.StyleId = 'line.staff.trill'))
{
obj.Duration = 0;
}
}
selection.RestoreSelection(); // now the original selection is restored.
score.Redraw = True;

****************************************
```

and the entire Run method of the plugin could look like

```
**********************************************
if (Sibelius.ScoreCount = 0)
{
Sibelius.MessageBox(_ScoreError);
return False;
}

score = Sibelius.ActiveScore;

if (score.StaffCount = 0)
{
Sibelius.MessageBox(_ScoreError);
return False;
}
```

```
fProcessEntireScore = False;
if (IsEmptySelection(score) = True)
{
fContinue = Sibelius.YesNoMessageBox(_msgSelectWholeScore); //True means Yes, continue and process score
if (fContinue = False)
{
return False; // they said no, so stop the plugin
}
fProcessEntireScore = True;
}

selection = score.Selection;
selection.StoreCurrentSelection();
score.Redraw = False; // this saves a lot of time avoiding redrawing each time the score changes

if (fProcessEntireScore = True)
{
selection.SelectPassage(1, score.SystemStaff.BarCount, 1, score.StaffCount);
}

for each obj in selection
{
if ((obj.IsALine) and (obj.StyleId = 'line.staff.trill'))
{
obj.Duration = 0;
}
}
selection.RestoreSelection(); // now the original selection is restored.
score.Redraw = True;
```

****

You also have to write IsSelectionEmpty() and your error messages and test that everything works, and fix it when (not if!) it doesn't.

Two final things I suggested to Ed (and which he includes in the published plugin):

1. this plugin will only work in Sib 5. If anyone tries to run it in Sib 4, it will fail when it gets to the Select() code. To avoid that, add a check at the very front that the version is at least 5.0:

****

```
if (Sibelius.ProgramVersion < zg_Sib5Version)
```

```
{
        MyMessageBox(_msgVersionTooEarly);
        return False;
}
**********************************************************
zg_Sib5Version is a global data item defined as

        zg_Sib5Version "5000"
```

(Note: I prefix normal globals with g_, globals used as dialog variables with dlg_, and globals I want to be able to find easily with zg_. Zg puts them at the end of the list of globals, so they are easier to update).

```
_msgVersionTooEarly is defined as

_msgVersionTooEarly "This plug-in requires Sibelius version 5.0 or later."
```

Messages like this, and strings in general, should be prefixed with _. This is a requirement for any plugins that Sib publishes – their translation tools look for the underscore. It is a bit tedious to put all strings in global data, but you get used to it, and if you do manage to sell a plugin to Sibelius, having it already set up saves a lot of work later.

The final thing I suggests was that Ed take is code that does the actual work, and put it in a separate method (called for example, ProcessSelection). This way, the Run method could be used unchanged as a template for a similar plugin. All you would need to change was the _PluginMenuName variable, the file name, and the contents of ProcessSelection.

So now Run would look like this.

```
**********************************
Run "() {
// Copyright 2009 Ed Hirschman. All rights preserved.

if (Sibelius.ProgramVersion < zg_Sib5Version)
{
        MyMessageBox(_msgVersionTooEarly);
        return False;
}

if (Sibelius.ScoreCount = 0)
{
        Sibelius.MessageBox(_ScoreError);
        return False;
}
```

```
score = Sibelius.ActiveScore;

if (score.StaffCount = 0)
{
	Sibelius.MessageBox(_ScoreError);
	return False;
}

fProcessEntireScore = False;
if (IsEmptySelection(score) = True)
{
	fContinue = Sibelius.YesNoMessageBox(_msgSelectWholeScore); // True means
Yes, continue and process score
   if (fContinue = False)
	{
		return False;  // they said no, so stop the plugin
	}
  fProcessEntireScore = True;
}

selection = score.Selection;
selection.StoreCurrentSelection();
score.Redraw = False; // this saves a lot of time avoiding redrawing each time the score
changes

if (fProcessEntireScore = True)
{
   selection.SelectPassage(1, score.SystemStaff.BarCount, 1, score.StaffCount);
}

ProcessSelection(score);

selection.RestoreSelection(); // now the original selection is restored.
score.Redraw = True;
}"
**********
```

Ed's plugin, which pretty much looks like this, is available for free download at http://www.sibelius.com/download/plugins/index.html?plugin=212.