Translating a possibly AI Generated Sibelius plugin to ManuScript

Bob Zawalich May 17, 2025

Contents

This was a post in the Facebook Sibelius Power users group from **Alisdair MacRae Birch** on May 14, 2025. Here is a link to the post:

https://www.facebook.com/groups/323691061147132/posts/2868792213303658/

Help/Advice Needed: How to Hide Text Elements in Multiple Scores - Sibelius Plugin?
I've been trying to write a plugin that processes a folder of Sibelius scores and hides specific text elements, including Title, Subtitle, Composer, Lyricist, Arranger, and Footer. I'm not well versed in sibelius plugins and encountering some challenges and would appreciate any guidance or advice. Is there another way? Can anyone help?

Here is the code, from another comment. I fixed the indentation but changed nothing else

```
PluginMethod Run()
{
// Prompt user for folder location
folderPath = Sibelius.FileBrowser("Select a folder containing Sibelius files", "", true);
if folderPath = "" then
{
    Sibelius.MessageBox("No folder selected. Operation cancelled.");
    return;
}
// Get list of .sib files in the folder
fileList = Sibelius.GetFileList(folderPath);
if Length(fileList) = 0 then
{
    Sibelius.MessageBox("No Sibelius files found in the selected folder.");
    return;
}
// Iterate through each Sibelius file
for i = 0 to Length(fileList) - 1
{
    sibFile = Sibelius.Open(fileList[i]);
    if sibFile = NULL then
    {
        Sibelius.MessageBox("Unable to open " & fileList[i]);
```

```
        continue;
    }
    textItems = sibFile.GetAllObjects("Text");
    for j = 0 to Length(textItems) - 1
    {
        if textItems[j].TextStyle = "Title" then
        {
            textItems[j].Visible = false;
        }
    }
    sibFile.Save();
    sibFile.Close();
}

Sibelius.MessageBox("Operation complete! Title text items are hidden in all files.");
}
```

It is clear to me that it is not written in ManuScript, though it is close. It may not have been AI generated, but I have no other guess for where it might have come from. I have seen other plugins that were generated by AI that used similar syntax.

Since this was short, I thought I might try translating it into ManuScript just to see if the algorithms worked.

Here are some of the deviations in this code from ManuScript. I am referring to these things as if I were looking at a .plg file in a text editor.

- PluginMethod Run()
    - A Run method would start with
        - **Run "() {**
    - And end with
        - **}"**
- If statement syntax, e.g., `if folderPath = "" then`
    - In ManuScript, an **if else** statement has the syntax

if else	`if (expression) block [ else block ]`
	for example:
	`if (found) {`
	`  Application.ShowFindResults(found);`
	`} else {`
	`  Application.NotFindResults();`
	`}`

        - The statement is an **if** enclosed in parentheses, and there is no "**then**" keyword

- These methods and variables do not exist or use different syntax in Sibelius, though similar ones are available:
    - Sibelius.FileBrowser
    - Sibelius.GetFileList
    - GetAllObjects
    - Length(fileList)
    - sibFile.Save();
    - sibFile.Close();
    - textItems[j].TextStyle
    - textItems[j].Visible

    - continue
    - Return needs a parameter, most commonly True or False
    - Sibelius.Open is close, but it takes a file name, not a file object as its parameter.

- The **Sibelius.MessageBox** instructions are correct.

So pretty much every line needs to be changed, but if we treat this as something written in a different language, it looks pretty reasonable.

Translating the plugin

I started with mechanical changes, such as making the **if** and **return** statements work. The changed text is marked in **bold** below.

I marked routines that do not exist or have incorrect syntax in red.

```
// Prompt user for folder location
folderPath = Sibelius.FileBrowser("Select a folder containing Sibelius files", "", true);
if (folderPath = "")
{
    Sibelius.MessageBox("No folder selected. Operation cancelled.");
    return False;
}
// Get list of .sib files in the folder
fileList = Sibelius.GetFileList(folderPath);
if (Length(fileList) = 0)
{
    Sibelius.MessageBox("No Sibelius files found in the selected folder.");
    return False;
}
// Iterate through each Sibelius file
for i = 0 to Length(fileList) - 1
{
    sibFile = Sibelius.Open(fileList[i]);
    if (sibFile = NULL)
    {
        Sibelius.MessageBox("Unable to open " & fileList[i]);
        continue;
    }
    textItems = sibFile.GetAllObjects("Text");
    for j = 0 to Length(textItems) - 1
    {
        if textItems[j].TextStyle = "Title" then
        {
            textItems[j].Visible = false;
        }
    }
    sibFile.Save();
    sibFile.Close();
}

Sibelius.MessageBox("Operation complete! Title text items are hidden in all files.");
```

Getting a folder containing scores

Here are the first 2 blocks of code

```
folderPath = Sibelius.FileBrowser("Select a folder containing Sibelius files", "", true);
if (folderPath = "")
{
   Sibelius.MessageBox("No folder selected Operation cancelled.");
   return False;
}

// Get list of .sib files in the folder
fileList = Sibelius.GetFileList(folderPath);
if (Length(fileList) = 0)
{
   Sibelius.MessageBox("No Sibelius files found in the selected folder.");
   return False;
}
```

Sibelius.SelectFolder is a direct replacement for **FileBrowser**. It returns a folder object rather than a path name, so it is checked for null rather than for an empty string.

```
folderMain = Sibelius.SelectFolder ("Select a folder containing Sibelius files");
if (folderMain = null)
{
   Sibelius.MessageBox("No folder selected. Operation cancelled.");
   return False;
}
```

Checking whether the folder has .sib files is similar to the original code

```
numFiles = folderMain.FileCount ("SIB");
if (numFiles = 0)
{
   Sibelius.MessageBox("No Sibelius files found in the selected folder.");
   return False;
}
```

Replacing the original code for these blocks, we now have this, where the new code is shown in green

```
// Prompt user for folder location

folderMain = Sibelius.SelectFolder ("Select a folder containing Sibelius files");
if (folderMain = null)
{
   Sibelius.MessageBox("No folder selected. Operation cancelled.");
   return False;
}

numFiles = folderMain.FileCount ("SIB");
if (numFiles = 0)
{
   Sibelius.MessageBox("No Sibelius files found in the selected folder.");
   return False;
}

// Iterate through each Sibelius file
for i = 0 to Length(fileList) - 1
{
   sibFile = Sibelius.Open(fileList[i]);
```

```
    if (sibFile = NULL)
    {
        Sibelius.MessageBox("Unable to open " & fileList[i]);
        continue;
    }
    textItems = sibFile.GetAllObjects("Text");
    for j = 0 to Length(textItems) - 1
    {
        if textItems[j].TextStyle = "Title" then
        {
            textItems[j].Visible = false;
        }
    }
    sibFile.Save();
    sibFile.Close();
}

Sibelius.MessageBox("Operation complete! Title text items are hidden in all files.");
```

Iterating through the Sibelius files

This code gets a single Sibelius file from the folder, and opens it if possible

```
// Iterate through each Sibelius file
for i = 0 to Length(fileList) - 1
{
    sibFile = Sibelius.Open(fileList[i]);
    if (sibFile = NULL)
    {
        Sibelius.MessageBox("Unable to open " & fileList[i]);
        continue;
    }
```

I will replace this with a normal way to do the same thing in ManuScript

Here is the syntax for getting files from a folder

Folder

Retrievable from methods of the Sibelius object.

`for each` *variable* **`in`** produces the Sibelius files in the folder, as **`File`** objects

`for each` *type variable* **`in`** produces the files of type in the folder, where type is a Windows extension.

- To get Sibelius score files (.sib), we can use **for each SIB sibFile in folderMain**. This will return a **File** object. Since we already checked for the presence of .sib file in the folder, this should never fail, and will return at least one .sib file.
- **Sibelius.Open** takes a file name (full path with extension), which we get from the **File** object sibFile. If **Open** fails, there is no **continue** command to skip the file as the original code does.
- To simulate the **continue** effect, I check whether **Open** succeeded. If not, I put up a message box, and nothing else will happen in the **For** loop. If it succeeded, the **else** clause is taken, and the score will be processed.

```
for each SIB sibFile in folderMain
{
```

```
if (Sibelius.Open (sibFile.NameWithExt, True) = False) // open quietly
{
   Sibelius.MessageBox("Unable to open " & sibFile.NameWithExt);
}
else // file was successfully opened. An opened file becomes the score object Sibelius.ActiveScore
{
```

Processing SystemTextItem objects in the opened score

```
textItems = sibFile.GetAllObjects("Text");
for j = 0 to Length(textItems) - 1
{
   if textItems[j].TextStyle = "Title" then
   {
      textItems[j].Visible = false;
   }
}
```

There is no **GetAllObjects** method. Instead, we need to get a **score** object. Rather oddly, the opened score is not returned by the **Open** command, but the variable **Sibelius.ActiveScore** is set to the score that was opened.

This sometime fails for no obvious reason. To keep the plugin from crashing by trying to process a null score, I added a check for score != null, and if null, the score will be skipped.

```
score = Sibelius.ActiveScore;
if (score != null)  // this sometime fails
{
```

Now that we have a valid score, we can look for **Title** text objects. We could look for them within a **Selection** if we had one, or in a **Staff** or in a **Bar**. Knowing that **Title** text is a **SystemTextItem** object, and that such objects live in the system staff, I do the equivalent of **GetAllObjects** like this:

```
for each SystemTextItem sText in score.SystemStaff // need to make a selection or look in a staff
 {
```

If any **SystemTextItem** objects are found, they will be returned one by one in the variable **sText**.

There are many kinds of **SystemTextItem** objects – see the **Text Styles** section of **Global Constants** in the **ManuScript Reference** for the list, names, and ids. These types are differentiated by the **StyleAsText** and **StyleId** fields of the **SystemTextItem** object.

For **Title** text **"Title"** is the **StyleAsText,** and **"text.system.page_aligned.title"** is the **StyleId.**I would normally use the **StyleId** here, since it is language-independent, but in order to stay closer top the original code I will use the **StyleAsText**. This means the plugin will only work in English.

```
   if (sText.StyleAsText = "Title")
   {
      sText.Hidden = True;
   }
}
```

OK, Deep breath. This is what we have now.

```
// Prompt user for folder location

folderMain = Sibelius.SelectFolder ("Select a folder containing Sibelius files");
if (folderMain = null)
{
   Sibelius.MessageBox("No folder selected. Operation cancelled.");
   return False;
}

numFiles = folderMain.FileCount ("SIB");
if (numFiles = 0)
{
   Sibelius.MessageBox("No Sibelius files found in the selected folder.");
   return False;
}

for each SIB sibFile in folderMain
{
   if (Sibelius.Open (sibFile.NameWithExt, True) = False) // open quietly
   {
      Sibelius.MessageBox("Unable to open " & sibFile.NameWithExt);
   }
   else // file was successfully opened. An opened file becomes the score object Sibelius.ActiveScore
   {
      score = Sibelius.ActiveScore;
      if (score != null)  // this sometime fails
      {
         for each SystemTextItem sText in score.SystemStaff // need to make a selection or look in a staff
         {
            // traces were used for debugging bz
            // trace("Run systext text, style as text: " & sText.Text & ", [" & sText.StyleAsText & "]");

            if (sText.StyleAsText = "Title")
            {
               //trace("title hidden: " & sText.Text);  // extra added by bobz for debugging
               sText.Hidden = True;
            }
         }

        sibFile.Save();
        sibFile.Close();
      }
}

Sibelius.MessageBox("Operation complete! Title text items are hidden in all files.");
```

Saving, Closing, and saying goodbye

So now we just need to save and close the score. These commands don't quite work.

```
sibFile.Save();
sibFile.Close();
```

This is what I will use instead:

```
score.Save();  // this is a Score, not File command
Sibelius.CloseAllWindowsForScore(score, False); // closes all open windows in the score. This is a Sibelius object command
```

There are lots of variants of Sibelius.Close, but this one will work the best here.

How there is just the final message to say we are done:

```
Sibelius.MessageBox("Operation complete! Title text items are hidden in all files.");
```

I changed this slightly to include the number of files processed and added a return statement. And we are done.

```
// added number of files processed to message bz
Sibelius.MessageBox("Operation complete! Title text items are hidden in " & numFiles & " files.");

return True;
```

Here is the fully translated plugin:

```
// Prompt user for folder location

folderMain = Sibelius.SelectFolder ("Select a folder containing Sibelius files");
if (folderMain = null)
{
   Sibelius.MessageBox("No folder selected. Operation cancelled.");
   return False;
}

numFiles = folderMain.FileCount ("SIB");
if (numFiles = 0)
{
   Sibelius.MessageBox("No Sibelius files found in the selected folder.");
   return False;
}

for each SIB sibFile in folderMain
{
   if (Sibelius.Open (sibFile.NameWithExt, True) = False) // open quietly
   {
      Sibelius.MessageBox("Unable to open " & sibFile.NameWithExt);
   }
   else // file was successfully opened. An opened file becomes the score object Sibelius.ActiveScore
   {
      score = Sibelius.ActiveScore;
      if (score != null)  // this sometime fails
      {
            for each SystemTextItem sText in score.SystemStaff // need to make a selection or look in a staff
         {
            // traces were used for debugging bz
            // trace("Run systext text, style as text: " & sText.Text & ", [" & sText.StyleAsText & "]");

            if (sText.StyleAsText = "Title")
            {
               //trace("title hidden: " & sText.Text);  // extra added by bobz for debugging
               sText.Hidden = True;
            }
         }

         score.Save();  // this is a Score, not File command
         Sibelius.CloseAllWindowsForScore(score, False); // closes all open windows in the score. This is a Sibelius object command
      }
   }
}

// added number of files processed to message bz
Sibelius.MessageBox("Operation complete! Title text items are hidden in " & numFiles & " files.");
```

```
return True;
```

This code will only hide Title text. You can get it to do more things by changing the code around the block

```
if (sText.StyleAsText = "Title")
{
   //trace("title hidden: " & sText.Text);  // extra added by bobz for debugging
   sText.Hidden = True;
}
```

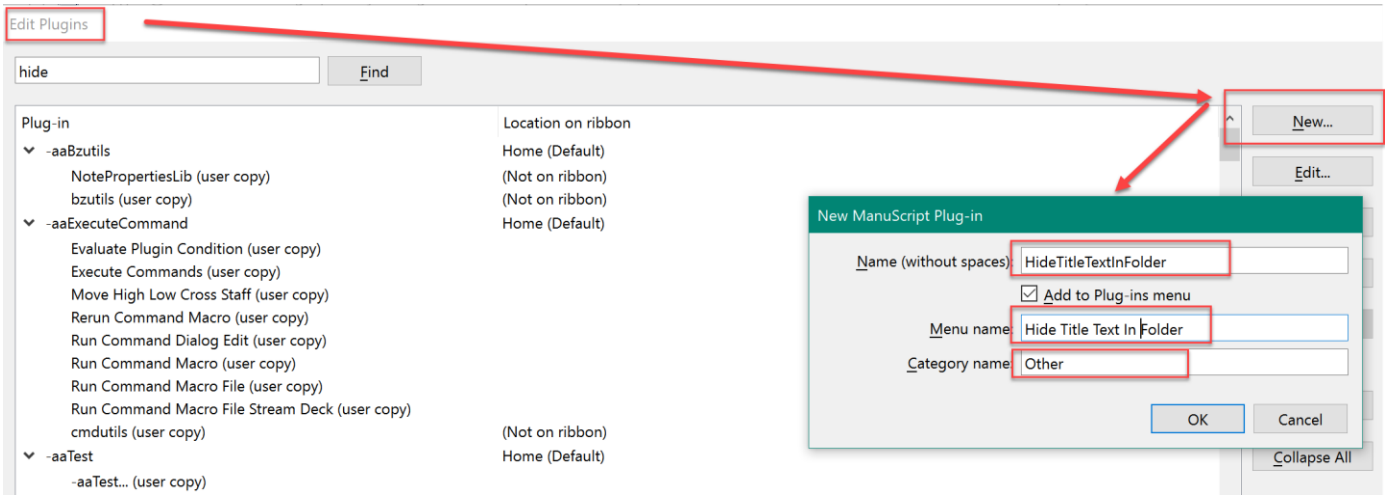
For example, to make it hide **Composer** and **Lyricist** text as well as title, you could check more **StyleAsText** values. If you do that as shown below, be sure to parenthesize each condition, since ManuScript's order of evaluation is left-to-right only, with no operator precedence. Watch that you spell the names exactly as defined and in the correct case, and that you are using plain, not curly, double quotes.

You might end up with this:

```
if ((sText.StyleAsText = "Title") or (sText.StyleAsText = "Composer") or (sText.StyleAsText = "Lyricist"))
{
   //trace("text hidden: " & sText.Text);  // trace added by bobz for debugging
   sText.Hidden = True;
}
```

Making a plugin that uses this code.

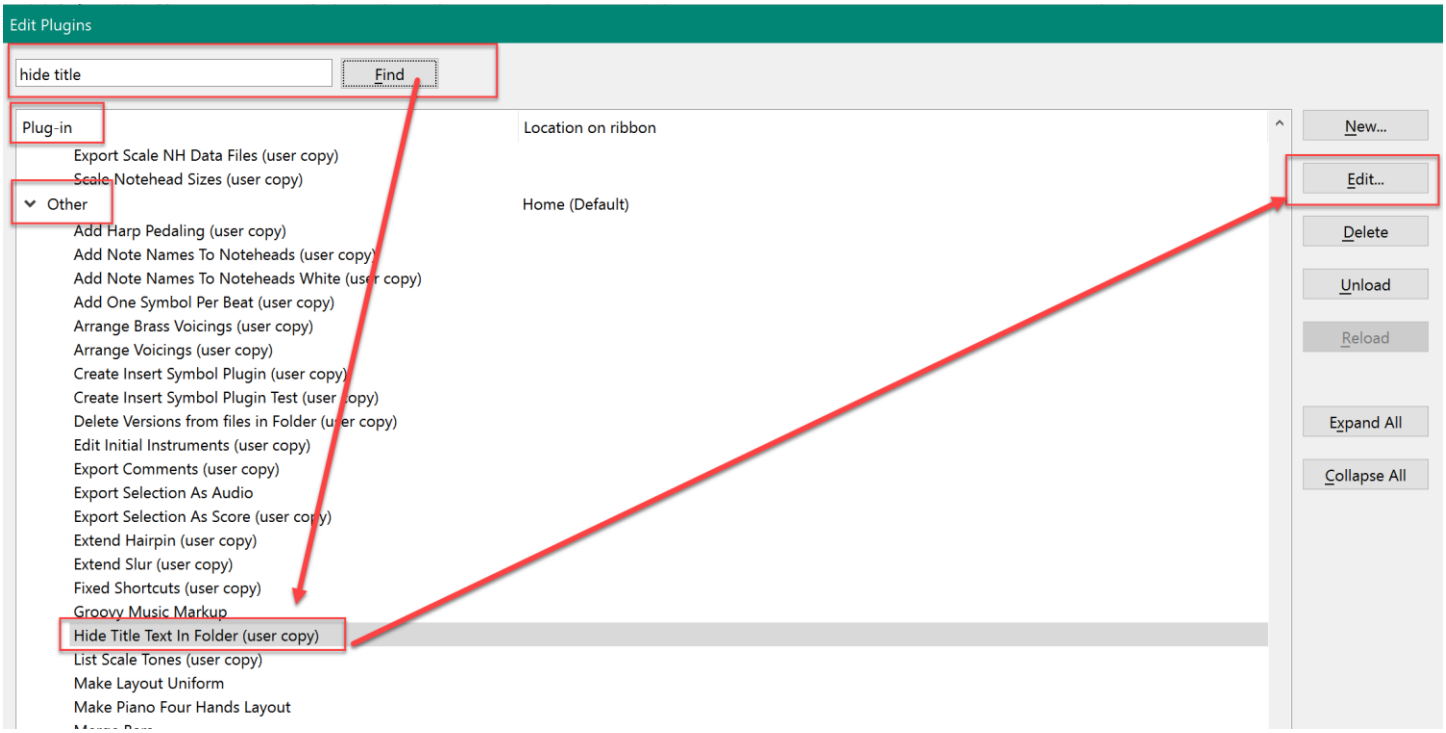
The easiest way to get to a working plugin it to go to **File>Plug-ins>Edit plug-in**, and press the **New…** button



It will ask for a (file) name, a menu name, and a **Category** name. The category is the name of a subfolder of the user plugins folder. "**Other**" will always be there, so it will work here.

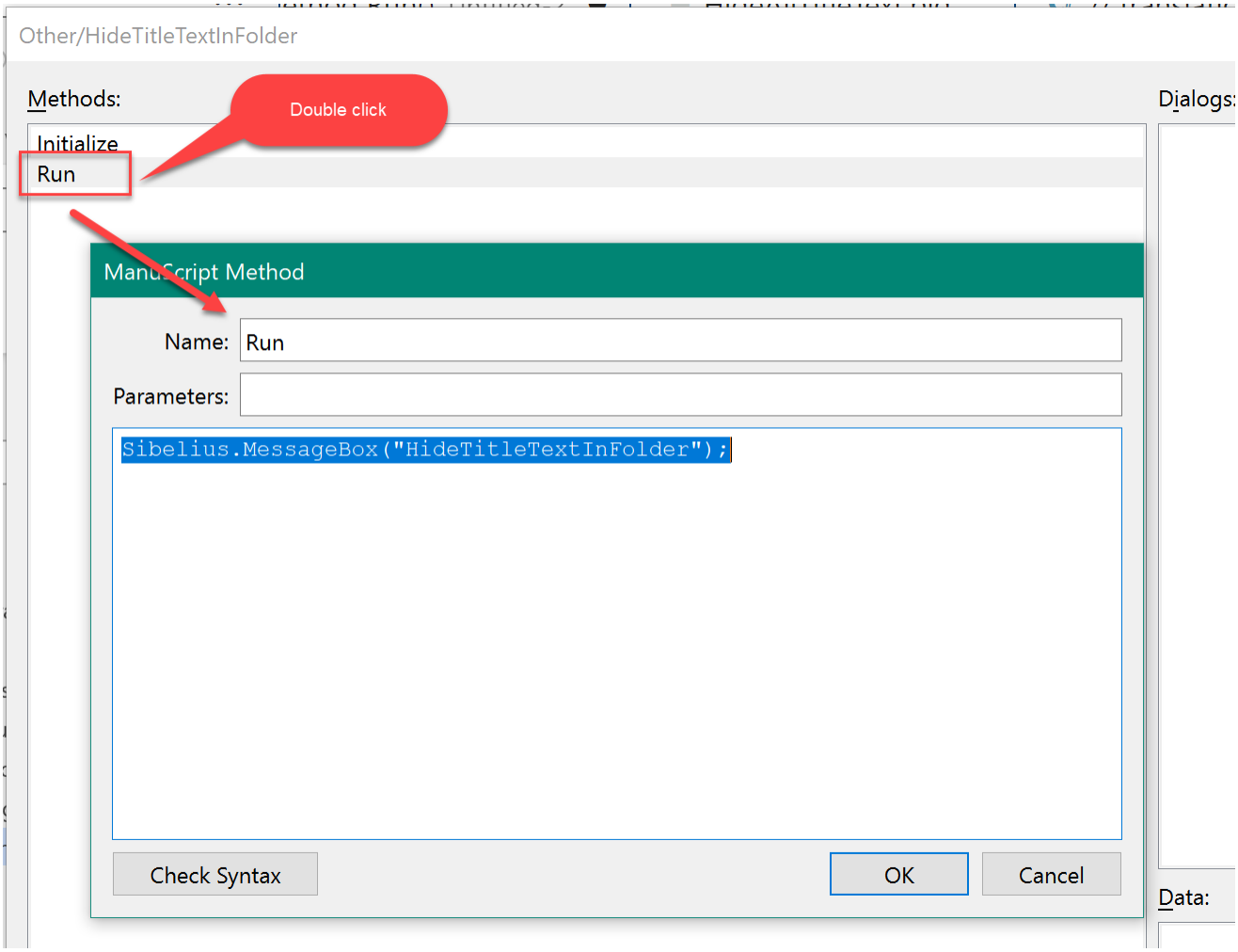
I am calling the file **HideTitleTextInFolder**. The file name should have no spaces. The menu name is what Sibelius will show you. I always make the menu name be the file name with spaces between the words.

Once you OK the **New** dialog, you will return to **Edit Plugins**. Look for **Other** in the list on the left, and inside **Other** look for **Hide Title Text In Folder**. You can use the Find box to make this easier to find. Select it and press the **Edit...** button.



This will open an editing window. Double click on the name Run, and it will open up a window editing the **Run** method.

You will see a **Sibelius.MessageBox** line. Press **Delete** to have an empty **Run**.



Now copy all the text of the translated plugin and paste it into the edit window. Press **Check Syntax**, which should show no errors. If there are errors, you will need to figure out how to correct them.

ManuScript Method

Name: Run

Parameters:

```
// Prompt user for folder location

folderMain = Sibelius.SelectFolder ("Select a folder containing Sibelius files");
if (folderMain = null)
{
    Sibelius.MessageBox("No folder selected. Operation cancelled.");
    return False;
}

numFiles = folderMain.FileCount ("SIB");
if (numFiles = 0)
{
    Sibelius.MessageBox("No Sibelius files found in the selected folder.");
    return False;
}

for each SIB sibFile in folderMain
{
    if (Sibelius.Open (sibFile.NameWithExt, True) = False) // open quietly
    {
        Sibelius.MessageBox("Unable to open " & sibFile.NameWithExt);
    }
    else // file was successfully opened. An opened file becomes the score object Sibelius.ActiveScore
    {
        score = Sibelius.ActiveScore;
        if (score != null)  // this sometime fails
        {
                for each SystemTextItem sText in score.SystemStaff // need to make a selection or look in a staff
            {
                // traces were used for debugging bz
                // trace("Run systext text, style as text: " & sText.Text & ", [" & sText.StyleAsText & "]");

                if (sText.StyleAsText = "Title")
                {
                    //trace("title hidden: " & sText.Text);  // extra added by bobz for debugging
                    sText.Hidden = True;
                }
            }

            score.Save();  // this is a Score, not File command
            Sibelius.CloseAllWindowsForScore(score, False); // closes all open windows in the score. This is a Sibelius
object command
        }
    }
}

// added number of files processed to message bz
Sibelius.MessageBox("Operation complete! Title text items are hidden in " & numFiles & " files.");

return True;
```

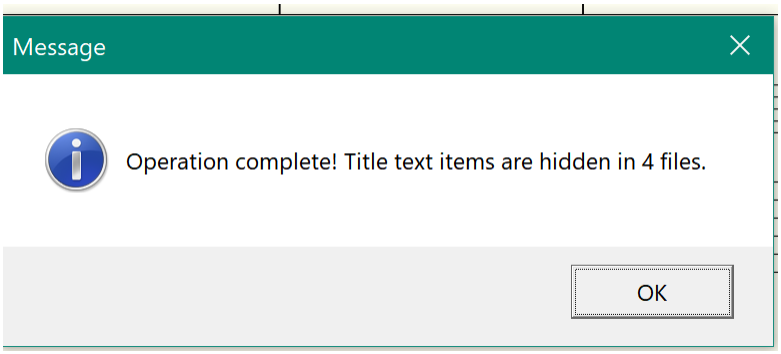
Check Syntax | OK | Cancel

Press **OK** once satisfied, then press **OK** or **Close** to close each dialog until you are back at the score.

To test, I suggest making a folder containing 2 or 3 scores. I would have a normal title in one score, a title using a wildcard in another, and no title in the 3rd.

Be sure that none of the scores in that folder are open before you run the plugin.

Run the plugin, which should give you a dialog to choose a folder. Choose your folder and the plugin will run. At the end you should see a message box like this:



There are lots of details this plugin will not cover well, and I would not publish a plugin in this state, but what we have is a pretty direct translation of the original plugin code. It should do the task it was asked to do.

Good luck!