Bob Zawalich Sibelius Variable Naming Conventions

Documented by Ed Hirschman July 2010

As a new plug-in writer, and someone who doesn't do programming for a living, I was looking for some coding guidelines that would help me to create plug-ins that are easy to understand, troubleshoot and maintain. I wasn't able to find quite what I was looking for, so I studied several of Bob Zawalich's plug-ins to derive some guidelines to follow. I have reviewed this with Bob and he has provided valuable additions and clarifications to what I was able to put together.

I am not implying that this is the best guideline there could be, but since Bob is the most prolific plug-in author, his style is the most prevalent out there for plug-ins not written by Sibelius staff. More importantly, it is logical and gets the job done. Please note that this scheme is not endorsed or sanctioned by Sibelius.

Prefixes for Global Variables

Variable Prefix Type	Examples	Used for	Note
dlg_Xxxx	dlg_lstAvailableParts dlg_fInstallPlugin dlg_strEditFolder	dlg_ for variables that are used in a dialog structure	
_Xxxxx	_ExistsText _PluginMenuName _captionSorting _msgNoneFound	Global text of a nature that would require translation for foreign language versions of Sibelius	A Sibelius standard. Variables starting with underscore are triggers to Sibelius to substitute in alternate foreign language text when used with foreign language versions. Be careful to follow this conversion for text to be translated from the start, so if Sibelius wants to buy your plugin and ship it, you don't need to rewrite it to comply. For underscore globals, the convention that existing plugins use is to have some descriptive text starting with a capital letter, such as _InitialText, or _PluginMenuName. _PluginMenuName might be improved from a naming standpoint if it was called as _namePluginMenu, but there are many precedents of this naming, so it is not worth worrying about.
g_Xxxx	g_fDoTrace g_strExistsText	Globals not in a dialog and not subject to translation.	Use g_ for other globals not covered by the above prefixes Types, and I try to minimize these. Often these will be things that dialog routines will need to know about, and you cannot pass parameters to such routines.
zg_ Xxxx	zg_PreferencesVersionNumber	A global I might need to change often, so it forces it to the end of the items in the dialog editor	A global I might need to change often, so alphabetical sorting forces it to the end of the items in the dialog editor Data panel.

Local & Global Variables

Functional Type**	Used for	Note
i	Index	Integer, usually starts at 0
arr	ARRay	
f	a boolean Flag (0/1 or true/false)	
str	STRing	Example, strNameInst - a string holding an instrument name
lst	for the contents of aLiSTbox orcombobox.	
nr	NoteRests	
n	notes	nStavesUsed
s	staff	
barnum	bar numbers	A Sibelius convention. Correct – barnumFirst Incorrect - "barNum", barNumFirst, firstBarNumber
val	for numerical VALue	– not for 0/1 (use f for that), these are for storing constants, not values that change e.g. valPi = 3.14
caption		for what would appear in a progress dialog box
msg	MeSsaGe	shown to user in a dialog box

Note and Guidelines for Variables

- Variable naming - Bob uses a form of "Hungarian", a technique developed by Charles Simonyi at Microsoft. He believed that good naming leads to better code, and that having a system for naming saves a lot of programmer work.
- Functional types" are in all lower case and are used to start the variable name, therefore all variables start with a lower case letter. The Functional type is usually followed by a descriptive noun, and possibly followed by an adjective. Examples:
    - "nStavesUsed"
    - "strNameInst", a string holding an instrument name.
- Names after the Functional Type begin with an upper case letter and follow Camel Casing convention.
- If the variable is only used locally in a small routine, the Functional Type may suffice for the variable name, such as using "i" as the variable name for the index in a "for" loop.
- These rules are frequently broken, by the way, especially when interacting with pre-existing code, but in general this is the model to use.
- If a new Functional Type is needed, feel free to create a new one.

Methods

- Method names all begin with capital letters, and are mainly unstructured descriptions of functions.
- Methods begin with an upper case letter and follow Camel Casing convention.
- Examples
    - BuildVersionText
    - CheckExistingFile
    - GetPreferences
    - MyYesNoMessageBox

Dialogs

- Dialogs all begin with capital letters, follow Camel Casing, and are mainly unstructured descriptions of the Dialog boxes.
- Dialogs end with the word "Dialog"
- Examples
    - FileExistsDialog
    - RestoreDialog
    - HelpDialog