

# **Visual TCL**

## **- vTcl -**

# **Technical Guide**

**- DRAFT -**

# vTcl Tech -Guide

---

## Table of contents

|                     |   |
|---------------------|---|
| 1 vTcl Widgets..... | 4 |
|---------------------|---|

# vTcl Tech -Guide

---

## Document scope

This document is not about using vTcl.

The scope of this document relates to maintainers and developers of the vTcl GUI building system. It explains the innards behind vTcl.

For Information on using vTcl in a session to draft a user program see the vTcl-User-Guide or the vTcl-Tutorial.

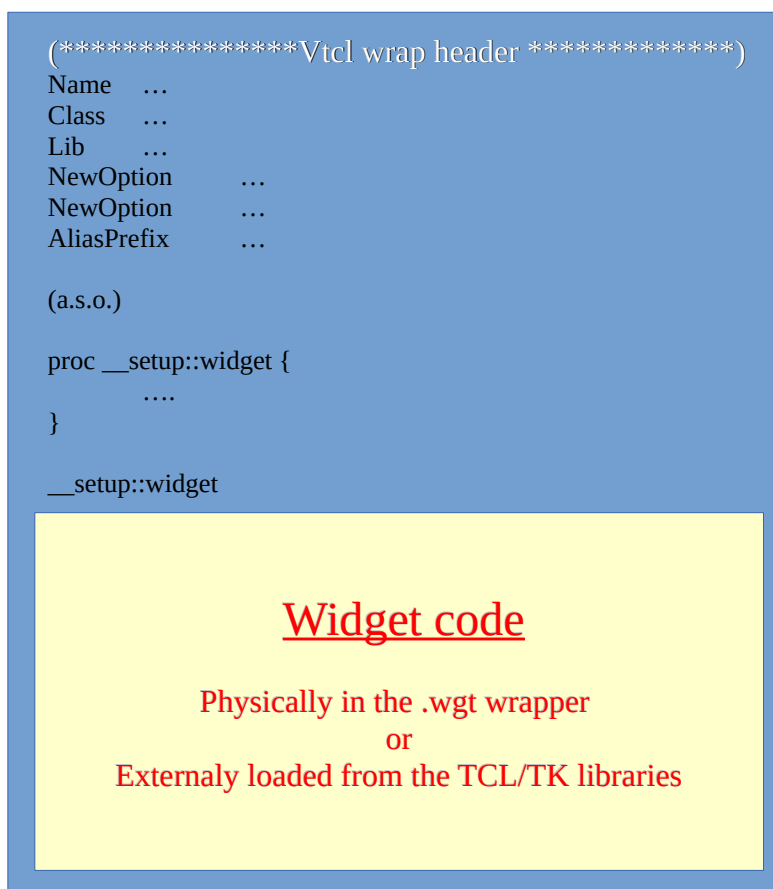
# vTcl Tech -Guide

## 1 vTcl Widgets

Widgets that are used in vTcl are not plain \*.tcl files.

In order to use a new widget type in vTcl it has to be wrapped in a widget definition file that links the widget options a.s.o to the vTcl GUI builder as a managable component of vTcl.

This is especially important for the vTcl Alias system that maps widget path names to comfortable alias names (→ Alias system, vTcl User-Guide)



This section explains how to define a widget in Visual Tcl. If you are making your own widgets for your install of VTcl, you should put your files in the 'user' subdirectory. This will ensure that you don't overwrite any VTcl widgets and that they will not conflict with later releases of VTcl.

You should be careful when defining widgets that you don't overwrite a default widget type or class. For example, don't name your widget button, because you will overwrite the default Tk button type. And, don't make a new button and call its Class 'Button', because again, you will overwrite the default button class. Instead, try to use your own names for widgets.

# vTcl Tech -Guide

Each widget is defined separately in its own file with the extension **.wgt**

The files live below `<vTcl-hook>/lib/Widgets/...` in a separate directory for the specific widget.

The following are the possible options in a widget file:

| WIDGET OPTION  | PARAMETER   | EXPLANATION  |
|----------------|---|--|
| Name           | <code>&lt;name&gt;</code>   | Name of Widget. This is also considered to be the "type" of widget by vTcl.  |
| Class          | <code>&lt;class name&gt;</code>   | Widget class   |
| IsSuperClass   | <code>&lt;yes   no&gt;</code>   | This widget is a super class widget. Super class widgets are widgets which are not created themselves, but represent sub widgets.<br><br>Example: scrollbars, scales. A scrollbar is a widget, but vTcl recognizes two different "types" of scrollbars: vertical and horizontal. So, Scrollbar is the super class, and scrollbar_v and scrollbar_h are the sub widget types. |
| SuperClass     | <code>&lt;class name&gt;</code>   | The super class of this widget. This widget is a subwidget.  |
| Lib            | <code>&lt;lib name&gt;</code>   | Widget library (core, itcl, tix, etc...)   |
| CreateCmd      | <code>&lt;command&gt;</code>  | Command to create the widget (defaults to <code>&lt;Class&gt;</code> )   |
| AddOptions     | <code>&lt;option&gt;..&lt;option&gt;..</code>                             | Options to <code>&lt;Command&gt;</code> when toolbar button is pressed.  |
| Icon           | <code>&lt;icon name&gt;</code>  | Toolbar icon (defaults to <code>icon_&lt;name&gt;.gif</code> )   |
| Balloon        | <code>&lt;text&gt;</code>   | Text of the toolbar balloon  |
| DefaultOptions | <code>&lt;option&gt; &lt;value&gt; [&lt;option&gt; &lt;value&gt;]*</code> | Options to <code>&lt;Command&gt;</code> as the widget is created.  |
| DefaultValues  | <code>&lt;option&gt; &lt;option&gt;</code>                                | Specifies that the given options will take their default value instead of the value given by the   |

# vTcl Tech -Guide

---

| WIDGET OPTION  | PARAMETER                                   | EXPLANATION  |
|----------------|---|--|
|                |   | option database at the time the widget is created  |
| DumpCmd        | <proc>                                      | Proc to dump the widget when saving (defaults to vTcl:dump_widget_top)   |
| DumpChildren   | <yes   no>                                  | Dump the widget's children when saving or pasting (yes or no)  |
| InsertCmd      | <proc>                                      | A command executed when the widget is created. (defaults to <name>)  |
| DoubleClickCmd | <proc>                                      | A command executed if the widget is double-clicked.  |
| TreeLabel      | <string   proc>                             | The string displayed next to the widget in the widget tree. If <string> has an @ as the first character, then string is the name of a proc to execute that will return the string to display. The proc must have one arguments: Widget<br>Widget is the widget to get the label for. |
| TypeCmd        | <proc>                                      | A proc to return what type a widget is. This must be specified if the widget is a super class. It tells vTcl how to determine which subclass widget it is looking at.<br><br>Example: scrollbar: horizontal or vertical?   |
| AutoPlace      | <yes   no>                                  | Ignore preference and automatically place the widget.  |
| Resizable      | <none   both   horizontal   veritcal>       | How the widget can be resized: none, both, horizontal, vertical  |
| Function       | <menu text> <command>                       | Adds a widget-specific function to the right-click menu.   |
| NewOption      | <option> <text> <type><br><choices> <title> | Adds a new option to the option list.  |
| Export         | <proc>                                      | This proc needs to be exported when a program is saved. When creating new widget types that are not standard, it is sometimes necessary to create procs to handle certain functions of the widget. By specifying the   |

# vTcl Tech -Guide

| WIDGET OPTION     | PARAMETER           | EXPLANATION  |
|-------------------|---------------------|--|
|                   |                     | proc as an Export, vTcl will export that proc to the save file when it saves, ensuring that the program will still run with the necessary procedures.  |
| WidgetProc        | <proc>              | This is the proc that widgets of this type will be aliased to when Command Aliasing is turned on.  |
| AliasPrefix       | <string>            | When auto-aliasing is turned on, widgets are automatically assigned an alias name based on AliasPrefix and the order of their creation. (defaults to <Class>)  |
| ResizeCmd         | <proc>              | <p>This is the proc that is called when a widget is grabbed by a handle and resized. vTcl takes care of whether or not the widget CAN be resized based on the &lt;Resizable&gt; argument. Once vTcl has determined that it can resize this widget, it calls this command.</p> <p>The proc must have three arguments:</p> <p>Widget Width Height</p> <ul style="list-style-type: none"><li>• Widget is the widget being resized</li><li>• Width is the new width for the widget</li><li>• Height is the new height for the widget</li></ul> <p>(default is vTcl:adjust_widget_size)</p> |
| AdditionalClasses | <class> <class> ... | Lists classes of children that vTcl should know of (eg. Notebook, Tabset, Page, ... for Iwidgets)  |

# vTcl Tech -Guide

---