Visual TCL

- vTcl -

Technical Guide

- DRAFT -

Table of contents

i. Revision history

Revision	Changes	Date / Token	State
0.1	Initial document	2016-02-12/hph	DRAFT
0.2a	Providing Widget definition section	2016-02-12/hph	-
0.2b	Started explanation of internal structures (Namespaces, vTcl private variables, …	2016-02-27/hph	-

ii. 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 of vTcl.

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

1 The vTcl concept …

2 Vtcl namespaces & internal variables

2.1 vTcl namespace

On startup vTcl establishes its private namespace in the global area.

namespace eval ::vTcl {}

2.2 vTcl associative array

vTcl(<name>,<subname>, ...)

2.3 vTcl internal procedures & functions

Vtcl uses internal names for its internal procedures & functions

proc vTcl:<pro/fun> { … }

These are mapped to its global namespace, since they start with 'vTcl'.

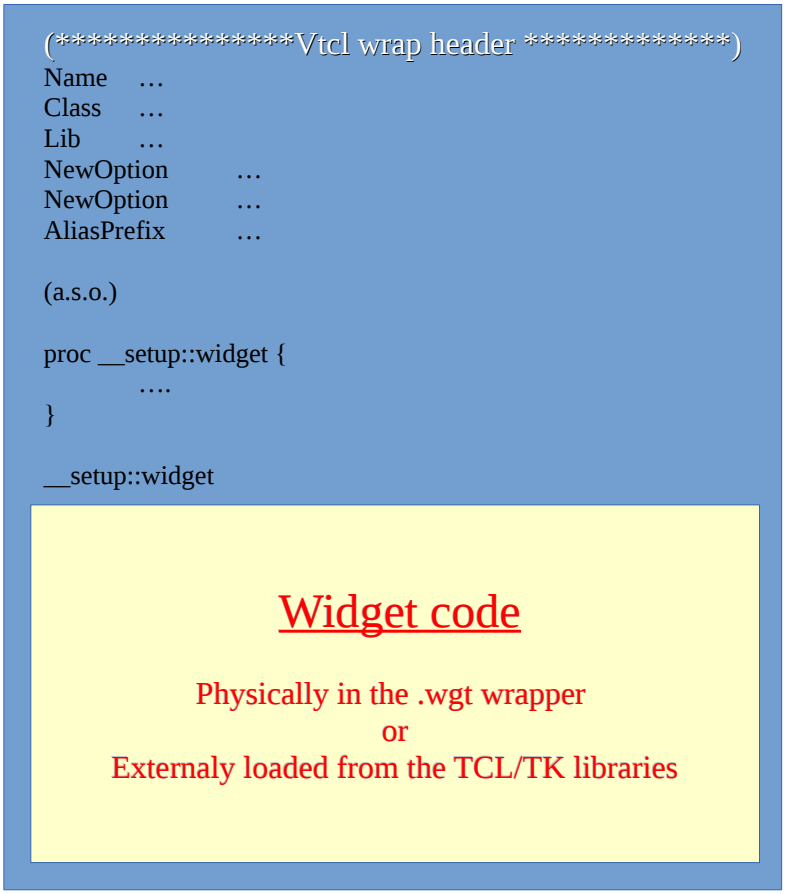
Other procedures ...

3 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 witdget 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.

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

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

3.1 Widget definition options

The following are the possible options in a widget file:

WIDGET OPTION	PARAMETER	EXPLANATION
Name	<name>	Name of Widget. This is also considered to be the "type" of widget by vTcl.
Class	<class name>	Widget class
IsSuperClass	<yes | no>	This widget is a super class widget. Super class widgets are widgets which are not created themselves, but represent sub widgets. 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	<class name>	The super class of this widget. This widget is a subwidget.
Lib	<lib name>	Widget library (core, itcl, tix, etc...)
CreateCmd	<command>	Command to create the widget (defaults to <Class>)
AddOptions	<option>..<option>..	Options to <Command> when toolbar button is pressed.
Icon	<icon name>	Toolbar icon (defaults to icon_<name>.gif)
Balloon	<text>	Text of the toolbar balloon
DefaultOptions	<option> <value> [<option>	Options to <Command> as the widget is

WIDGET OPTION	PARAMETER	EXPLANATION
	<value>]*	created.
DefaultValues	<option> <option>	Specifies that the given options will take their default value instead of the value given by the 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 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. 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> <choices> <title>	Adds a new option to the option list.

WIDGET OPTION	PARAMETER	EXPLANATION
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 necesary to create procs to handle certain functions of the widget. By specifying the 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 necesary 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>	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 <Resizable> argument. Once vTcl has determined that it can resize this widget, it calls this command. The proc must have three arguments: Widget Width Height • Widget is the widget being resized • Width is the new width for the widget • Height is the new height for the widget (default is vTcl:adjust_widget_size)
AdditionalClasses	<class> <class> ...	Lists classes of children that vTcl should know of (eg. Notebook, Tabset, Page, ... for Iwidgets)

Appendix