



**2.20**

**Documentation**

# TKproE 2.20 Documentation

## Table of Contents

About TKproE.....	5
TKproE License.....	5
Other license information.....	6
Introduction.....	7
Learn more about TCL/TK at:.....	7
Installation.....	8
Command line start up options.....	8
Synopsis.....	8
Description.....	8
Options.....	8
application_file_name .....	8
application options .....	8
The TKproE development environment.....	9
TKproE manipulates a running application.....	9
Benefits of this approach.....	9
Disadvantages of this approach.....	9
TKproE uses one interpreter for both programs.....	9
Some names are reserved for use by TKproE.....	9
The developer has full access to TCL/TK.....	9
Overview.....	10
Menu bar.....	10
Action button bar.....	10
Widget tool bar.....	10
Themed widget bar.....	11
Message bar.....	11
Current widget path.....	11
TKproE Menu Bar Options.....	12
Action Button Bar.....	15
Save.....	15
Properties.....	15
Packer.....	15
Placer.....	15
Gridder.....	15
Bindings.....	15
Procedures.....	15
Global Parameters.....	16
Widget tree.....	16
Cut.....	16
Copy.....	16
Paste.....	16
Trash.....	16
TKproE general options dialog.....	16
Auto save interval.....	17
Auto save directory.....	17
Ask for widget name on insertion.....	17
Default geometry manager.....	17
Project type.....	17
Include PostgreSQL support.....	17

# TKproE 2.20 Documentation

Editor.....	17
PDF viewer.....	18
Global variables dialog.....	18
Watch Window.....	19
Namespace Exclusion List.....	19
Procedures.....	21
Special procedures.....	22
Selecting the current working widget.....	22
Importance of the current widget path.....	22
Selecting and displaying Toplevel windows.....	23
Inserting widgets into the application.....	24
Configuring Toplevel Windows.....	24
Automatic window management.....	24
Window properties.....	25
Configuring widgets.....	26
Standard widget properties dialog box.....	26
The Canvas Widget.....	27
Selecting the current canvas object.....	28
Adding a canvas object.....	28
Specifying canvas object tags.....	28
Adjusting canvas object coordinates.....	28
Moving an object.....	28
New coordinates.....	28
Other canvas object properties.....	28
Using the draw tools.....	29
Attach.....	29
Copy.....	29
Cut.....	29
Detach.....	29
GrpMove.....	29
ID.....	30
Lower.....	30
ObjMove.....	30
OneMove.....	30
Raise.....	30
Stretch.....	30
None.....	30
The Paned Window Widget.....	31
The Notebook Widget.....	32
Menu configuration.....	34
Widget bindings.....	35
Geometry Management.....	36
The Placer.....	36
The Packer.....	38
The Gridder.....	39
Font Selection and Management.....	41
Application widget tree.....	42
Icon Library.....	43
Image Management.....	44
Additional Widget Support.....	45

# TKproE 2.20 Documentation

Advanced Features.....	46
Templates.....	46
Source modules and Namespaces.....	46
The generated code.....	47
TPstartupSrc and TPendSrc procedures.....	47
Image definitions.....	47
Named fonts.....	47
Variable initialization procedure.....	47
Toplevel creation procedures.....	47
User defined procedures.....	48
Invocation of TPstartupSrc.....	48
Invocation of TPinitVars procedure.....	48
Display the toplevel windows.....	48
Invocation of TPendSrc procedure.....	48
Converting existing TCL/TK applications to TKproE.....	49

# TKproE 2.20 Documentation

## About TKproE

Programmer: Dennis R. LaBelle

TKproE is an integrated program development environment for the TCL/TK scripting language. TKproE, itself, is completely written in the TCL/TK language.

TKproE supports the rapid development of sophisticated graphical user interfaces. TKproE takes advantage of TK, a widget set that is accessible through TCL, a very efficient interpreted programming language. With TKproE the user can build or modify application programs while they are running. This makes it possible to test program changes immediately without the cost of compiling the application.

## TKproE License

Copyright (c) 2004-2016 by Dennis R. LaBelle (labelled@nycap.rr.com) All Rights Reserved.

This software is provided 'as-is', without any express or implied warranty. In no event will the authors be held liable for any damages arising from the use of this software.

Permission is granted to anyone to use this software for any purpose, including commercial applications, and to alter it and redistribute it freely, subject to the following restrictions:

1. The origin of this software must not be misrepresented; you must not claim that you wrote the original software. If you use this software in a product, an acknowledgment in the product documentation would be appreciated but is not required.
2. Altered source versions must be plainly marked as such, and must not be misrepresented as being the original software.
3. This notice may not be removed or altered from any source distribution.

# TKproE 2.20 Documentation

## Other license information

TKproE borrows existing source code from several sources. To meet license terms and Copyright notice requirements associated with some of this code the following information is provided from the original source code distributions.

---

### Notice related to source code of procedures:

TP\_ImageEncode

TP\_ImageDecode

---

This software is copyrighted by Ajuba Solutions and other parties. The following terms apply to all files associated with the software unless explicitly disclaimed in individual files.

The authors hereby grant permission to use, copy, modify, distribute, and license this software and its documentation for any purpose, provided that existing copyright notices are retained in all copies and that this notice is included verbatim in any distributions. No written agreement, license, or royalty fee is required for any of the authorized uses. Modifications to this software may be copyrighted by their authors and need not follow the licensing terms described here, provided that the new terms are clearly indicated on the first page of each file where they apply.

IN NO EVENT SHALL THE AUTHORS OR DISTRIBUTORS BE LIABLE TO ANY PARTY FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OF THIS SOFTWARE, ITS DOCUMENTATION, OR ANY DERIVATIVES THEREOF, EVEN IF THE AUTHORS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

THE AUTHORS AND DISTRIBUTORS SPECIFICALLY DISCLAIM ANY WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT. THIS SOFTWARE IS PROVIDED ON AN "AS IS" BASIS, AND THE AUTHORS AND DISTRIBUTORS HAVE NO OBLIGATION TO PROVIDE MAINTENANCE, SUPPORT, UPDATES, ENHANCEMENTS, OR MODIFICATIONS.

GOVERNMENT USE: If you are acquiring this software on behalf of the U.S. Government, the Government shall have only "Restricted Rights" in the software and related documentation as defined in the Federal Acquisition Regulations (FARs) in Clause 52.227.19 (c) (2). If you are acquiring the software on behalf of the Department of Defense, the software shall be classified as "Commercial Computer Software" and the Government shall have only "Restricted Rights" as defined in Clause 252.227-7013 (c) (1) of DFARS. Notwithstanding the foregoing, the authors grant the U.S. Government and others acting in its behalf permission to use and distribute the software in accordance with the terms specified in this license.

---

### Notice related to use of Silk Icons developed by Mark James.

The Silk Icons, developed by Mark James, are included in TKproE in a base64 encoded format. This set of icons is used under the [Creative Commons Attribution 2.5 License](https://creativecommons.org/licenses/by/2.5/). You can visit the Silk Icons home page at [www.famfamfam.com/lab/icons/silk](http://www.famfamfam.com/lab/icons/silk).

# TKproE 2.20 Documentation

## Introduction

TKproE is a program development environment for the TCL/TK scripting language. TKproE, itself, is completely written in the TCL/TK language.

TKproE capabilities include:

1. Support for all TCL/TK 8.6 widgets.
2. Image management.
  - The option of automatically including image files as base64 encoded strings.
  - The ability to preview loaded images.
3. Namespace management
  - The ability to save namespaces in separate files.
  - The ability to include or exclude specific namespaces when saving your project.
4. Font management
  - A font selection tool with visual examples
  - Named font support
5. Text widget support
  - Saving and restoration of text widget content.

## Learn more about TCL/TK at:

[TCL/TK on-line documentation](#)

[Tcl Developer Xchange](#)

[TCLers' Wiki](#)

[Active State](#)

# TKproE 2.20 Documentation

## Installation

TKproE installation is performed by extracting the distribution archive into a directory of your choosing. The subdirectory structure within the archive is expected by TKproE and should be maintained.

You will need the TCL/TK windowing shell (wish) installed on your system in order to run TKproE.

## Command line start up options

### *Synopsis*

```
wish tkproe.tcl [tkproe options]
```

```
tkproe options: [application_file_name [application options]]
```

### *Description*

TKproE allows the user to interactively build and modify a graphical user interface, based upon the TCL/TK scripting language. It provides full access to the complete command and widget set of TCL/TK version 8.6.

User-generated errors (e.g., invalid TCL syntax) that occur during the use of TKproE are displayed in a pop-up window for the user to acknowledge. Untrapped errors generated by TKproE itself are printed to stderr.

### *Options*

The following command line options are recognized by TKproE.

#### [application\\_file\\_name](#)

The first option must be the name of the main application file or, if the file does not exist, the name of the new application. The loading of any other additional TCL files is the responsibility of the main application.

#### [application options](#)

All command line options following the main application file name are passed to the main application as parameters.



# TKproE 2.20 Documentation

## The TKproE development environment

The following key facts should be understood about creating a software application using TKproE.

***TKproE manipulates a running application.***

### Benefits of this approach

- While developing, the application looks and behaves exactly as the user will experience it.
- There is no compilation or additional processing required to produce the final application.
- It reduces the amount of data that has to be used to store the contents of the currently developed application. The program under construction itself contains all information which is necessary to create a *TCL/TK* file containing the program definition

### Disadvantages of this approach

- Writing bad code can be fatal. For example, writing then executing an infinite loop would result in a complete freeze up of the development environment. In such a case, the developer would be unable to regain control of the program in order to save the application to file.

## ***TKproE uses one interpreter for both programs***

TKproE and the application to be built are part of the same execution environment. TKproE variables, procedures and widgets are differentiated from the application under development by naming convention. When saving an application under development, TKproE simply doesn't save those portions of the programming environment that are part of TKproE itself.

## ***Some names are reserved for use by TKproE***

Some names are reserved for the components that comprise TKproE.

TKproE imposes a certain application framework upon the source code that it generates for an application. As a result of this, some procedure names are reserved for use by TKproE when it generates code.

Therefore, the following rules must be followed when writing an application with TKproE:

- Procedure names cannot start with the string **TP\_**
- Toplevel window names cannot start with the string **.tp**
- Global variable names cannot start with the string **TP**

## ***The developer has full access to TCL/TK***

New widgets are created with reasonable default parameters, to prevent the need for changing every newly created widget. Nevertheless, every aspect of the application can be changed within TKproE. If a widget parameter cannot be changed from TKproE's standard dialogs it is always possible to modify the application directly at the WISH console.

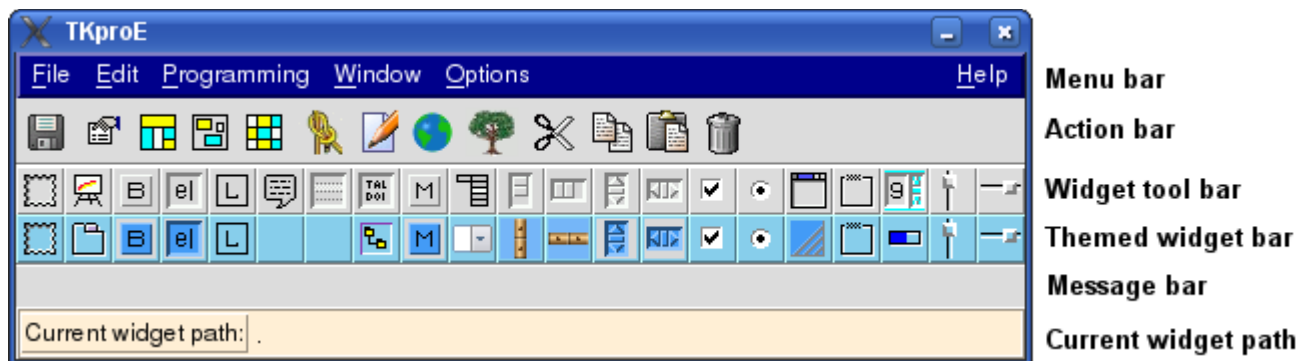
Additionally, a text editor may be used to make any desired changes to the source code generated by TKproE. The modified source code file can then be opened by TKproE.

TCL/TK scripts can also be SOURCED into TKproE using the Source option of the File menu.

# TKproE 2.20 Documentation

## Overview

After starting TKproE , two toplevels appear on the screen. The empty toplevel window is the workspace where the application is to be built. This is the WISH root window. The second window is the Main *TKproE* dialog.



The main dialog contains the following six basic sections.

### **Menu bar**

The menu bar contains drop-down menus that may be used to access various TKproE features.

### **Action button bar**

The Action Button Bar contains several buttons that can be used to perform the most frequent tasks.

### **Widget tool bar**

The widget tool bar contains a button for each of the widgets that may be inserted into your application. The buttons are used to insert the following TK widgets.



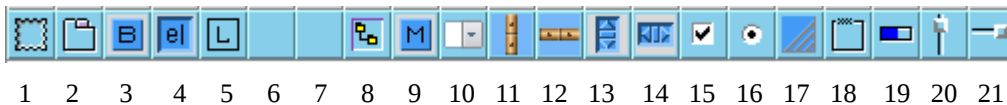
1. frame
2. canvas
3. button
4. entry box
5. label
6. message box
7. list box
8. text box
9. menu button
10. menu
11. vertical scale
12. horizontal scale
13. vertical scrollbar
14. horizontal scrollbar
15. check box

# TKproE 2.20 Documentation

16. radio button
17. toplevel window
18. labeled frame
19. spin box
20. horizontal paned window
21. vertical paned window

## ***Themed widget bar***

The themed widget tool bar contains a button for each of the new themed widgets provided by TCL/TK version 8.5. These widgets may also be inserted into your application. The buttons are used to insert the following themed TK widgets.



1. frame
2. notebook
3. button
4. entry box
5. label
6. blank – not used
7. blank – not used
8. treeview
9. menu button
10. combobox
11. vertical separator
12. horizontal separator
13. vertical scrollbar
14. horizontal scrollbar
15. check box
16. radio button
17. sizegrip button
18. labeled frame
19. progress bar
20. horizontal paned window
21. vertical paned window

## ***Message bar***

The message bar consists of a normally blank line that is used to display messages from TKproE.

## ***Current widget path***

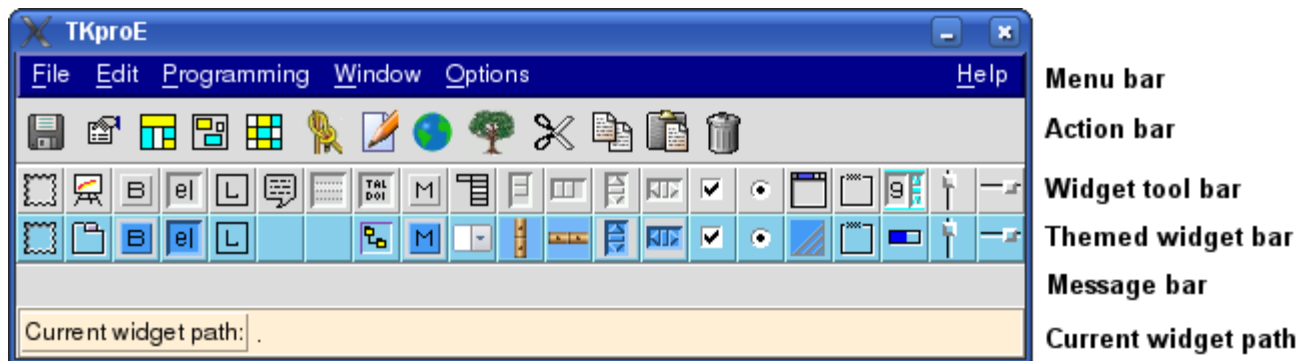
The currently selected widget is displayed on the last row of the TKproE main dialog. It is found to the right of a button labeled as the [Current widget path](#). Almost all operations performed by TKproE's dialog boxes are performed on the widget identified by the current widget path. The current widget path can be copied to the clipboard by pressing mouse button 3 while the mouse is hovering over the widget path.

Pressing the **Current widget path** button displays a list of all toplevel windows currently defined for the application under development. On this list, window names that have a checked box near them

# TKproE 2.20 Documentation

are currently displayed. The user may check the box to display the associated window or uncheck the box to hide the window.

## TKproeE Menu Bar Options



**Figure:** Main TKproE dialog

The options available on the TKproE menu bar are described below.

File	
New	This option clears out the current application without saving it first. TKproE will ask for verification prior to completing this action.
Open	Use this option to load a different application. This action will clear out the current application before loading the new one. Save the current application before opening a new one. The name of the current file being edited is displayed in the title bar of the TKproE main window.
Source	Use this option to perform a [source] command on a TCL script file. This will add the procedures found in the TCL script file to the current application under development.
Save	Use this option to save the application currently under development. The application will be saved using the current file name for the project. This file name is set when either the file is opened by the command line, the <b>Open</b> menu option is used or the <b>Save As</b> menu option is selected.
Save As	Use this option to specify the file name that should be used to save the current application. Once the name is specified TKproE will write the application to the specified file and the file name will appear on the title bar of the TKproE main window.
Save Namespace	Use this option to save any one of the namespaces contained within the current application to a file.
Quit	Use this option to exit TKproE. TKproE will prompt the user to verify this action is really desired but will not save the current application under development. Any file saves must be done prior to using the <b>Quit</b> option.

# TKproE 2.20 Documentation

<b>Edit</b>	
<a href="#">Cut</a>	This option places a copy of the widget pointed to by the <a href="#">current widget path</a> into a buffer then deletes the widget. The widget can be as simple as a single button or as complex as a toplevel window along with all of its content.
<a href="#">Copy</a>	This option places a copy of the widget pointed to by the <a href="#">current widget path</a> into a buffer. The widget can be as simple as a single button or as complex as a toplevel window along with all of its content.
<a href="#">Paste</a>	This option pastes any widgets stored in the buffer into the widget specified by the <a href="#">current widget path</a> .
<a href="#">Delete</a>	This option deletes the widget pointed to by the current widget path. The widget can be as simple as a single button or as complex as a toplevel window along with all of its content.
<a href="#">Clear cutbuffer</a>	This option empties the buffer used to cut, copy or paste widgets within TKproE.
<a href="#">Save cutbuffer</a>	This option saves the contents of the cutbuffer to a TCL script file that can be used as a template to insert the stored widgets into any TKproE project.
<a href="#">Load template</a>	This option loads the cutbuffer with widgets previously saved to a template file.
<a href="#">Insert template</a>	This option loads the cutbuffer with the contents of a widgets template file then pastes the buffer into the location specified by the <a href="#">current widget path</a> .

<b>Programming</b>	
<a href="#">TPstartupSrc</a>	This option will display the <a href="#">TKproE procedures</a> dialog selected to the <b>TPstartupSrc</b> procedure. TKproE organizes the code it generates such that the <b>TPstartupSrc</b> procedure is executed as the first code in the application
<a href="#">TPendSrc</a>	This option will display the <a href="#">TKproE procedures</a> dialog selected to the <b>TPendSrc</b> procedure. TKproE organizes the code it generates such that the <b>TPendSrc</b> procedure is executed after all toplevel windows have been displayed.

<b>Window</b>	
This menu provides a list of important TKproE dialog boxes that can be quickly opened or closed.	
Fonts	When this item is checked the <a href="#">TKproE font selection dialog</a> is displayed. This dialog can be used to create, delete or modify named fonts.
Global variables	When this item is checked the <a href="#">TKproE global variables dialog</a> is displayed. This dialog can be used to create, delete or modify global variables for any TCL Namespace.
Icon library	When this item is checked the <a href="#">TKproE icon library dialog</a> is displayed. This dialog can be used to quickly add icons (small images) to your application.

# TKproE 2.20 Documentation

<b>Window</b>	
This menu provides a list of important TKproE dialog boxes that can be quickly opened or closed.	
Images	When this item is checked the <a href="#">TKproE image management dialog</a> is displayed. This dialog can be used to add any TK-supported image type to your application.
Procedures	When this item is checked the <a href="#">TKproE procedures dialog</a> is displayed. This dialog can be used to create, delete or modify TCL procedures for your application.
Toplevel windows	When this item is checked the <a href="#">TKproE toplevel windows dialog</a> is displayed. This dialog can be used to open or close any toplevel window of your application.
Watch window	When this item is checked the <a href="#">TKproE parameter watch dialog</a> is displayed. This dialog allows you to monitor the value of any TCL variable within your application.
Widget tree	When this item is checked the <a href="#">TKproE application widget tree dialog</a> is displayed. This dialog displays a list of all TK objects (e.g., buttons, toplevel windows) within your application. Double-clicking on an item in the list will set the current widget path to the selected item.
Console	When this item is checked the TCL/TK console window is displayed (if available on your operating system).

<b>Options</b>	
<a href="#">Reset widget counter</a>	TKproE maintains a sequential counter to help automatically generate unique names when adding new widgets. The addition of a new widget increments the counter by one. This counter may be reset to zero by selecting this option.
<a href="#">General</a>	This option displays the <a href="#">TKproE general options dialog</a> .

<b>Help</b>	
<a href="#">About</a>	This option displays the about box which contains the TKproE revision level, revision date and credits.
<a href="#">TKproE documentation</a>	<p>When this option is selected TKproE attempts to open the TKproE documentation with a PDF viewer program.</p> <p>TKproE will first attempt to use the PDF viewer defined in the <a href="#">TKproE general options dialog box</a>. If that program cannot be located then TKproE will:</p> <ol style="list-style-type: none"><li>1. Under the Windows operating system, use the default PDF reader to display its documentation.</li><li>2. Under Linux, attempt to use Acrobat Reader (the acroread program). If this fails, TKproE attempts to use the kghostview program.</li></ol>

# TKproE 2.20 Documentation

## Action Button Bar



The Action Button Bar can be found on the Main TKproE dialog. It consists of a icon buttons that can be used to perform some of the most frequent actions necessary when building a TCL/TK application. The individual buttons are described below. Pressing one of these buttons with mouse button 1 will display a dialog or perform an action against the widget identified by the Current widget path.

For some of the buttons, a shortcut is available to simultaneously select a new current widget and then execute an action button. The mouse cursor turns into a question mark pointer when it is on top of one of these buttons. The shortcut is performed with the following steps:

1. Press and hold mouse button 3 with the mouse cursor over the desired action button.
2. Move the mouse pointer over the widget you wish to affect.
3. Release mouse button 3.



### **Save**

Pressing this icon will save the application currently under development. The application will be saved using the current file name for the project. This file name is set when either the file is opened by the command line, the **Open** menu option is used or the **Save As** menu option is selected.



### **Properties**

Pressing this icon will display a dialog that can be used to configure the widget identified by the Current widget path. In most cases, this will be the general Widget properties dialog. However, in some cases (e.g., canvas widgets) a more specific dialog will be displayed.



### **Packer**

This icon displays the Packer geometry manager dialog.



### **Placer**

This icon displays the Placer geometry manager dialog.



### **Gridder**

This icon displays the Gridder geometry manager dialog.



### **Bindings**

This icon displays the Widget bindings dialog.



### **Procedures**

This icon displays the Procedures dialog.

# TKproE 2.20 Documentation



## **Global Parameters**

This icon displays the Global variables dialog.



## **Widget tree**

This icon displays a dialog listing all the widgets currently displayed by the user application. The current widget path may be changed by double-clicking on the name of one of the widgets in the list.



## **Cut**

Pressing this icon will copy the current widget to a cutbuffer then delete the widget.



## **Copy**

Pressing this icon will copy the current widget to a cutbuffer.



## **Paste**

Pressing this icon will copy any widget in the cutbuffer into the current widget path.



## **Trash**

Pressing this icon will delete the current widget.

## TKproE general options dialog

TKproE general options

Auto save interval: 10 minutes

Auto save directory: /share/pdisk1/tkprogs/autosave Browse

Ask for widget name on insertion: ☒

Default geometry manager: ☒ packer ☐ placer ☐ gridder

Project type: ☐ TCL only ☒ TCL/TK

Editor: kwrite {FILENAME}

PDF viewer: /usr/bin/okular

Done Save

The following general options can be specified and saved to the **TPconfig.txt** file that TKproE reads when it first starts up. The **TPconfig.txt** file is only read if it is found in the same directory as the TKproE program.

Press the **Save** button to save your options to file.



# TKproE 2.20 Documentation

Press the **Done** button to close the dialog box.

## ***Auto save interval***

This option is used to specify the interval, in minutes, between automatic saves of the current application under development. Set the value to zero to prevent TKproE from doing any automatic backups of the application.

## ***Auto save directory***

This option is used to specify the location of the subdirectory that should be used to store TKproE's automatic backups of the current application.

## ***Ask for widget name on insertion***

Check this option to force TKproE to ask the user for a new widget name when adding a TK widget. Uncheck this option if TKproE should automatically assign new widget names without asking.

## ***Default geometry manager***

This option selects the default geometry manager that TKproE will use when inserting new widgets.

## ***Project type***

TKproE can be used to create TCL-only as well as combined TCL/TK applications. TKproE will not attempt to save TK components for projects identified as "TCL only". In "TC"/TK" mode, both TCL and TK components will be saved.

## ***Editor***

This option specifies the external editor that should be used for editing the application procedures. This editor is run by pressing the Edit button on the [TKproE procedures dialog](#). The currently selected procedure will be opened using the external editor. The TKproE application itself will wait until the external editor program is closed before TKproE will respond to any more user input. When the editor is closed, TKproE will update the source code associated with the currently selected procedure.

Remember to save the file before exiting the external editor. You should also remember to press the **Insert** button in TKproE in order to save the modified procedure within TKproE.

External editors (e.g., Windows Write) that create a copy of themselves and then immediately return to the calling application will not work with this feature. (On the Windows operating system, the Notepad program will work properly.)


Any necessary command line options should be included when setting the value of this option. The name of the temporary file containing the procedure being edited will be substituted anywhere the word FILENAME is found in the option value.

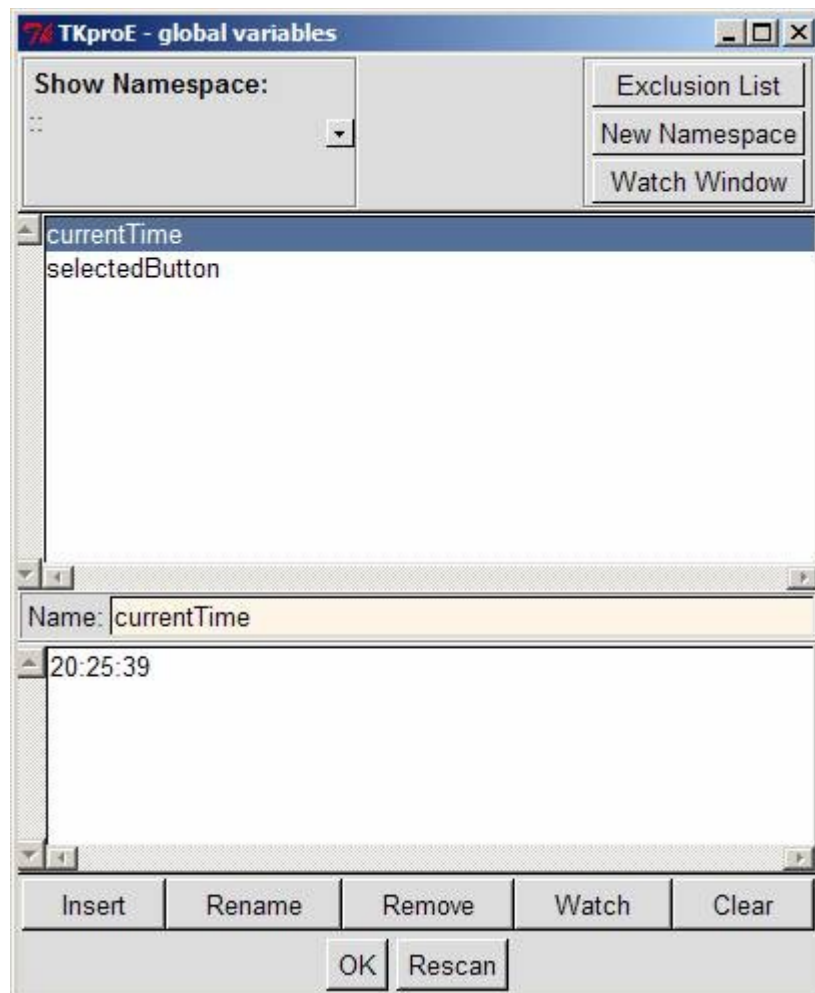
## ***PDF viewer***

This option is used to specify the path to the application that should be used for viewing TKproE's PDF formatted documentation.

# TKproE 2.20 Documentation

## Global variables dialog

TCL variables can be created, deleted and configured using the dialog shown below. This dialog is displayed by pressing the  button on the [Action Button Bar](#).



Global variables are administered separately for each Namespace known to the user application. The current Namespace is selected using a drop-down menu in the upper left section of the dialog.

TKproE makes the variables in some Namespaces (e.g., `::tcl` and `::tk`) read-only. These Namespaces are placed on an exclusion list that can be displayed using the **Exclusion List** button in the upper right hand side of the dialog. Additional Namespaces may be added to the exclusion list by the user.

A new Namespace can be easily created by pressing the **New Namespace** button

Existing variables are displayed in the listbox in the top half of the dialog. Selecting a variable in this listbox will display its value in the bottom half of the dialog. This value can be edited then re-saved by pressing the **Insert** button.

A new variable can be added by entering its name in the middle of the dialog, entering a value in the bottom half of the dialog then pressing the **Insert** button.

Press the **Rename** button to change the name of the currently selected variable.

# TKproE 2.20 Documentation

Press the **Remove** button to delete the currently selected variable.

Press the **Clear** button to blank out the name and value boxes.

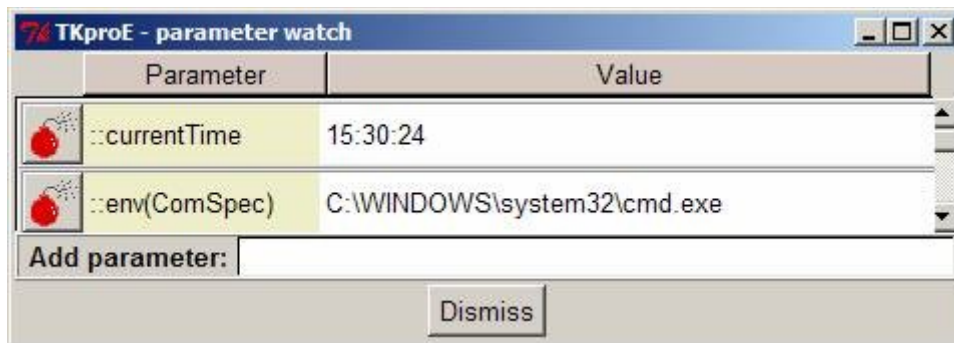
If a variable is currently highlighted, pressing the **Watch** button will display the Watch window and add the highlighted variable to the list of parameters that are displayed in it.

The list of existing variables can be updated by pressing the **Rescan** button. This may be necessary if new variables are created by the user application after the dialog is opened.

Press the **OK** button to close the dialog.

## Watch Window

The window shown below can be used to continuously display the current value of any TCL variable within your application. This dialog is displayed by pressing the **Watch Window** button on the [Global variables dialog](#). The dialog may also be displayed by selecting *Window/Watch Window* from the [Menu bar](#).



New parameters are added to the Parameter watch window by pressing the **Watch** button on the [Global variables dialog](#) while the variable is selected. A variable may also be added by typing its name in the **Add parameter** entry box then pressing the Enter key.

Watched parameters are removed by pressing the button immediately to the left of the parameter name.

Remember, it is not possible to unset a variable while it is being watched since the variable is “in use”.

## Namespace Exclusion List

Namespaces loaded into an application are automatically included in the source code generated by TKproE. However, some namespaces (e.g., **::tcl** and **::tk**) should never be saved. Therefore, TKproE has a default list of namespaces that will not be saved. The user may add to this list with the help of a special TKproE dialog box.

This dialog box may be displayed from either the *TKproE procedures* or the *TKproE global variables* dialog boxes. A namespace can be added to the exclusion list using a drop down menu button on the lower right hand side of the dialog.

# TKproE 2.20 Documentation



**Figure:** TKproE Namespace Exclusion List dialog

An item is removed from the list by pressing the right mouse button while the mouse pointer is positioned over the namespace. This will display a single menu item that can be selected to remove the namespace from the exclusion list.




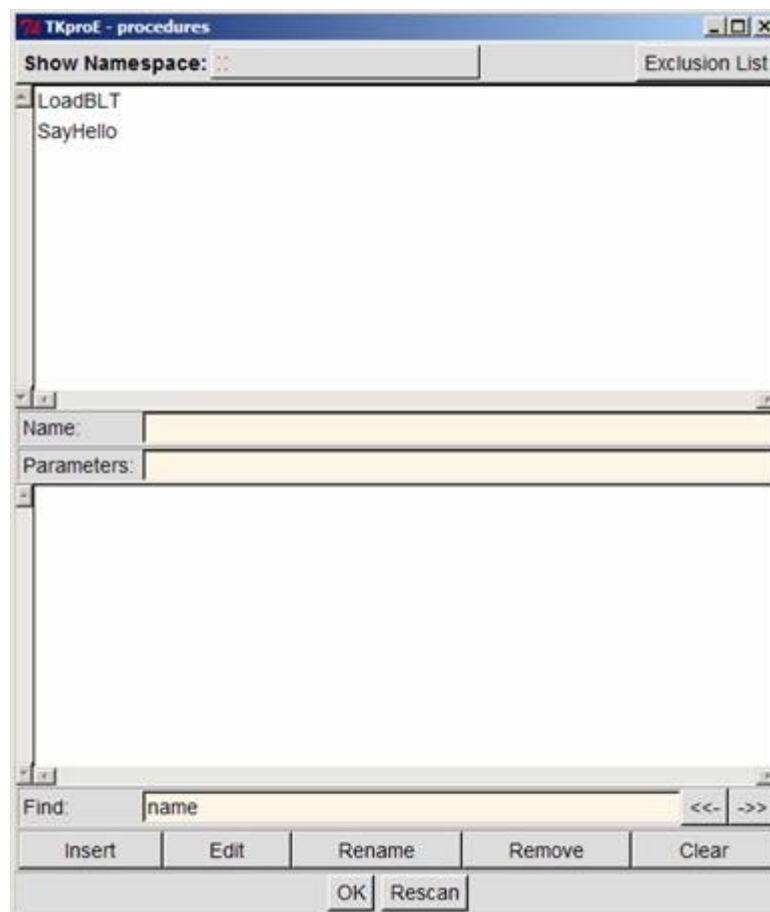
**Figure:** Removing a namespace from the exclusion list

Grayed-out items cannot be removed from the list since they represent the namespaces on TKproE's default list.

# TKproE 2.20 Documentation

## Procedures

TCL procedure development is supported with the *TkproE procedures* dialog. This dialog is activated by selecting the Windows/Procedures option from the TKproE menu bar. From this dialog procedures can be created, modified, deleted or renamed. The dialog may also be displayed by pressing the  [action button](#).



**Figure:** TKproE procedures dialog

The list of existing procedures is displayed by namespace. By default, the procedures available in the global (::) namespace is listed. A drop-down menu from which any of the existing namespaces can be selected is found in the upper left hand section of the dialog. Selecting any of these namespaces will display the associated procedures for that namespace. The procedures for certain namespaces (e.g., ::TCL, ::TK) have been made read-only. Such namespaces are found on the [namespace exclusion list](#) that may displayed using the button in the upper right of the dialog. TKproE allows the user to add other namespaces to this list.

The body of the currently selected procedure can be searched for a user specified string. To do so, enter the find string then press the ->> button to move the cursor forward to the next occurrence of the string. Press the <<- button to move the cursor backward to the previous occurrence of the string.

To edit a procedure, select its name from the list found in the upper half of the *TKproE procedures* dialog. This will display the procedure's name, parameters and source code in the lower half of the dialog. After changes have been made to the parameters and source code, press the Insert button to make the changes permanent.

# TKproE 2.20 Documentation

The currently selected procedure may be edited using an external editor defined by the [TKproE general options dialog](#) by pressing the **Edit** button. The TKproE application itself will wait until the external editor program is closed before TKproE will respond to any more user input. When the editor is closed, TKproE will update the source code associated with the currently selected procedure. External editors (e.g., Windows Write) that create a copy of themselves and then immediately return to the calling application will not work with this feature. (On the Windows operating system, the Notepad program will work properly.)

The **Rename** button allows the user to rename the currently selected procedure.

The **Remove** button allows the user to delete the currently selected procedure.

Pressing the **Clear** button is a quick method of blanking out the display of the name, parameters and source code text in preparation for creating a new procedure.

The **ReScan** button should be pressed if procedures have been added or deleted by activities outside the *TKproE procedures* dialog while the dialog has been open. **SOURCEing** an external file is an example of such a situation.

Press the **OK** button to close the dialog.

## ***Special procedures***

In addition to user-written procedures, the development environment also contains special procedures that are used by TKproE. The procedure dialog only gives access to procedures that the user is allowed to change. The special TKproE procedures (they have names that start with TP) are hidden from the user.

However, there are two special procedures that the user is allowed to edit. They are named **TPstartupSrc** and **TPendSrc**. To modify these procedures, select the menu items **Programming/TPstartupSrc** and **Programming/TPendSrc**. This will display the *TKproE procedures* dialog selected to the **TPstartupSrc** or **TPendSrc** procedure.

If the user application needs control when the application program is started, the code should be added to the **TPstartupSrc** procedure. TKproE generates the user application source code such that this procedure is the first code executed.

If the application needs control after the toplevel windows (containing the dialog components) have been displayed, the code should be added to the **TPendSrc** procedure. TKproE generates the user application source code such that this procedure is called as the last code executed prior to entering the TK event loop.

## **Selecting the current working widget**

### ***Importance of the current widget path***

The currently selected widget is displayed on the last row of the TKproE main dialog. It is found to the right of a button labeled as the **Current widget path**. Almost all operations performed by TKproE's dialog boxes are performed on the widget identified by the current widget path.

The current widget path can be set to any visible widget by pressing and holding mouse button 3 while the mouse is on the **Current widget path** button, then releasing the mouse button with the mouse pointer over the desired widget. When mouse button 3 is released, the displayed widget path will be updated.

# TKproE 2.20 Documentation

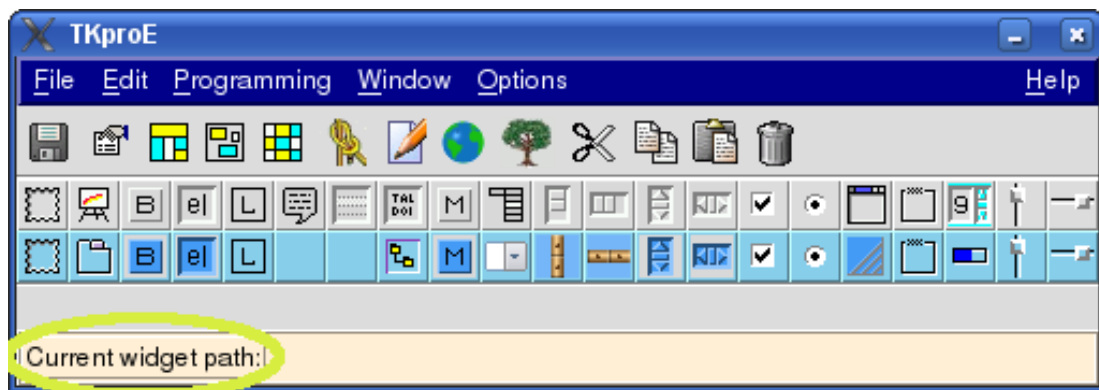
The widget path is also a menu bar

The dot separators in the displayed widget path also serve as drop-down menus that list other sibling widgets at the same level of the currently displayed subpath. Selecting a sibling from the drop-down menus will change the current path to be that of the sibling.

Clicking any part of the displayed path, other than the separating dots, will truncate the current widget path up to the selected part.

## Selecting and displaying Toplevel windows

It may be necessary to first display a toplevel window in order to select one of its widgets as the current widget path.



Pressing the **Current widget path** button displays a pop-up window containing a list of all toplevel windows currently defined for the application under development.



When this pop-up opens TKproE builds [TPopenWindow](#) and [TPcloseWindow](#) procedures, if necessary, for all currently displayed toplevel windows. A toplevel window is considered “defined” when TKproE has constructed **TPopenWindow** and **TPcloseWindow** procedures for it. These procedures are normally generated when TKproE is used to create a new toplevel window. Pressing the **Rescan** button will force TKproE to build **TPopenWindow** and **TPcloseWindow** procedures for any toplevel windows that have been created since the pop-up was opened.



# TKproE 2.20 Documentation

## Note

Opening the Toplevel windows dialog is a good way to make TKproE recognize the existence of toplevel windows when converting a legacy application to a TKproE based project.

On the window list, window names that have a checked box near them are currently displayed. The user may check the box to display the associated window or uncheck the box to hide the window. Checking the box executes the **TPopenWindow** procedure for the window while unchecking executes the **TPcloseWindow** procedure.

## Inserting widgets into the application

Application building generally consists of inserting new widgets and specifying the TCL code that should be executed when events (e.g., mouse clicks) occur.

Widgets are inserted into the application by selecting the desired object from the one of the widget toolbars. This will display a small dialog window asking the user to enter the new widget's name. A unique name is already provided in the dialog box and can be either accepted or replaced. New widgets are inserted into the [current widget path](#).

## Configuring Toplevel Windows

### *Automatic window management*

Toplevel windows (including the main TK window ".") contain the widgets that form an application's graphical interface. The main TK window "." is the root of the widget tree. An application usually contains various dialog components implementing different aspects of the program. They are normally displayed depending on the current status of the program. It is therefore important to be able to hide/show toplevel windows.

The main TK window "." can be hidden with the command: `[wm state . withdrawn]`. To display the window, the command: `[wm state . normal]` is used. This also works for the other toplevel windows, but TKproE provides an additional way to show/hide the toplevel windows.

Your application code should be written to display toplevel windows by calling the automatically created procedure **TPopenWindow.<toplevelName>**. The string <toplevelName> should be replaced with the name of the toplevel window. For example, the window **.myaboutbox** would be displayed with a call to the **TPopenWindow.myaboutbox** procedure. This procedure creates the toplevel window from scratch.

To remove a window, the procedure **TPcloseWindow.<toplevelName>** should be called. This procedure completely destroys the toplevel window. When the application is running within the TKproE environment the **TPcloseWindow.<toplevelName>** procedure also updates the **TPopenWindow.<toplevelName>** procedure before destroying the window.

When TKproE saves the application, the current display status of the toplevel windows is saved. This means that when the application is started, the toplevel windows that were displayed when the program was saved are displayed, and the toplevel windows that were hidden when the program was saved are not displayed. To change the display status of a window when developing with

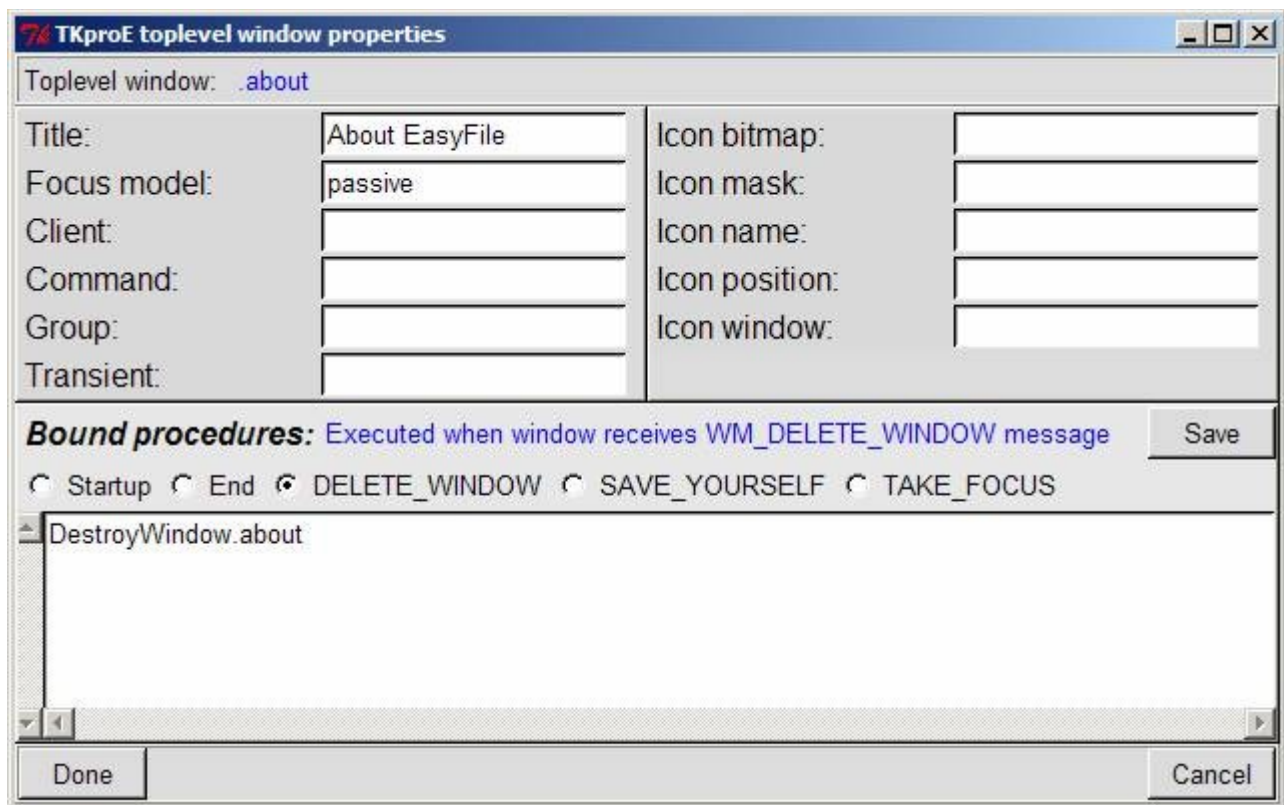


# TKproE 2.20 Documentation

TKproE, press the **Current widget path** button. This will display a list of all toplevel windows currently defined for the application under development. On this list, window names that have a checked box near them are currently displayed. The user may check the box to display the associated window or uncheck the box to hide the window.

## Window properties

Properties specific to a toplevel window itself may be set using the dialog shown below. This dialog box can be displayed by pressing the **More** button when the *Widget properties* dialog is displayed for the toplevel window.



The image shows a dialog box titled "TKproE toplevel window properties". At the top, it says "Toplevel window: .about". Below this, there are two columns of labels and text entry fields. The left column contains: "Title:" (with "About EasyFile" entered), "Focus model:" (with "passive" entered), "Client:", "Command:", "Group:", and "Transient:". The right column contains: "Icon bitmap:", "Icon mask:", "Icon name:", "Icon position:", and "Icon window:". Below these fields, there is a section titled "Bound procedures:" followed by the text "Executed when window receives WM\_DELETE\_WINDOW message". To the right of this text is a "Save" button. Below the "Bound procedures:" section, there are five radio buttons: "Startup", "End", "DELETE\_WINDOW" (which is selected), "SAVE\_YOURSELF", and "TAKE\_FOCUS". Below the radio buttons is a scrollable text area containing the text "DestroyWindow.about". At the bottom of the dialog, there are two buttons: "Done" on the left and "Cancel" on the right.

The toplevel window properties of greatest interest are normally the procedures that TKproE automatically binds to key toplevel window events. These events occur:

- Immediately prior to toplevel window creation (Startup event)
- Immediately after toplevel window creation (End event)
- When the window receives a message from the operating system to delete itself
- When the window receives a message from the operating system to save itself
- When the window receives a message from the operating system to take the focus

The properties dialog allows you to enter and save some TCL code that will be executed when the event occurs. Simply select the radiobutton for the event of interest, type in some TCL code into the scrollable text box then press the **Save** button.


Press the **Cancel** button to discard your changes and close the toplevel window properties dialog.

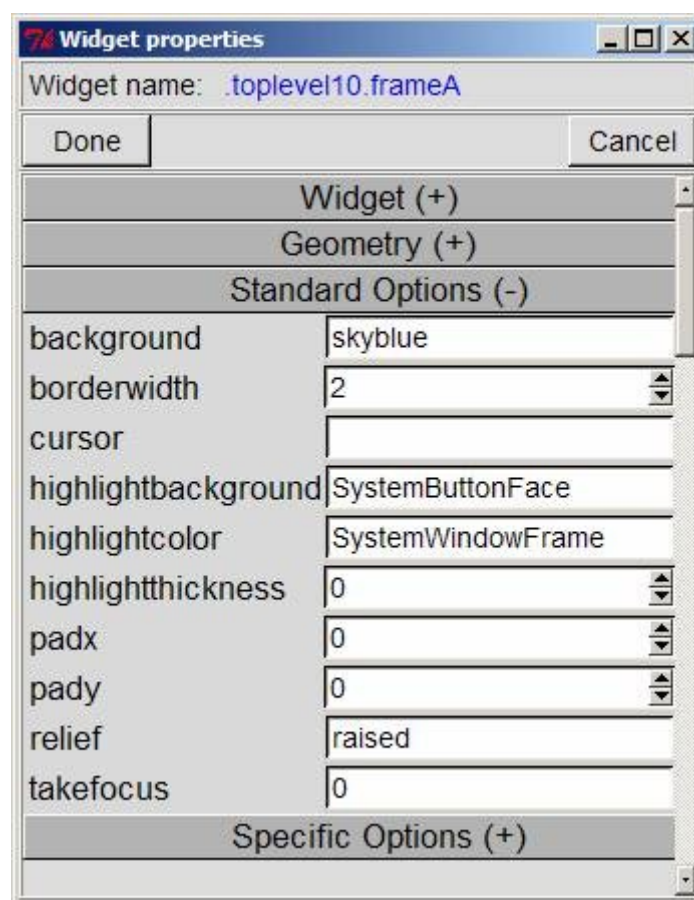
Press the **Done** button to accept your changes and close the toplevel window properties dialog.

# TKproE 2.20 Documentation

## Configuring widgets

### *Standard widget properties dialog box*

There are many configurable widget properties. Some properties are standard and apply to almost all widgets while others are specific to the class of widget. These properties are normally configured using TKproE's *Widget properties* dialog. This dialog can be displayed by pressing the  [action button](#). In this case, the dialog can be used to change the properties of the [current widget](#) that was selected at the time the dialog was opened. The dialog can also be displayed by shift-clicking with mouse button 3 on the widget to be configured. This activates the *Widget properties* dialog for modification of the shift-clicked widget.



**Figure:** Widget properties dialog

The *Widget properties* dialog contains four expandable sections containing properties that are automatically displayed based on the class of widgets being configured. Press mouse button 1 on the section header to expand or contract the section. Pressing mouse button 3 on some of these properties will display a special drop-down menu or dialog that allows an interactive selection of possible values.

A fifth expandable section will be available when configuring one of the themed widgets. This section will be labeled **States** and allows you to set the state properties of a themed widget.

New values for properties take effect as soon as either:

# TKproE 2.20 Documentation

- They are selected from a drop-down menu or special dialog
- The Enter key is pressed.
- Focus is moved to another property
- The section containing the property is contracted.
- The properties dialog is closed.

A very important property is the *command* property. This property allows you to bind functionality (TCL commands) to a widget (e.g., a button or a menu item).

Press the **Cancel** button to cancel all property changes for the widget.

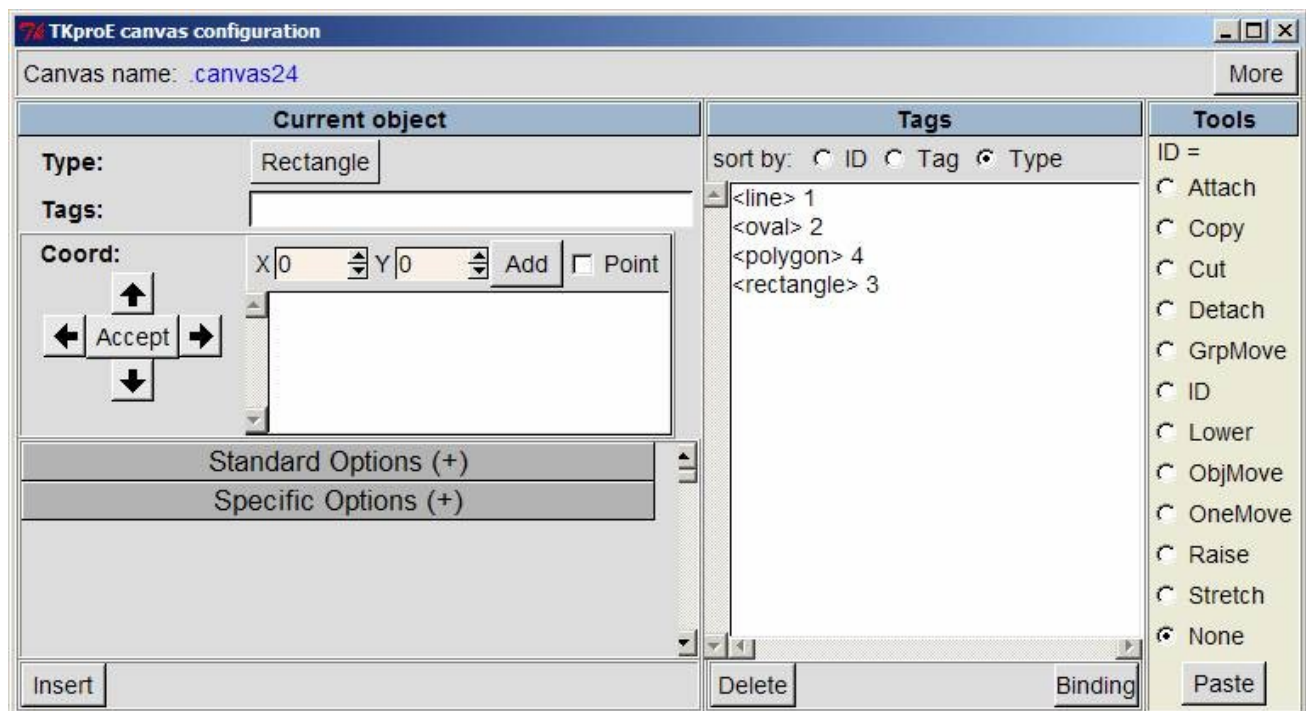
Press the **Done** button to close the *Widget properties* dialog.

Most widget classes share the common *Widget properties* dialog. However, some widget classes have their own properties dialog. These special dialog boxes provide support for specialized features of the widget class (e.g., drawing graphical objects in a canvas widget). These special dialogs are

automatically displayed in response to the  [action button](#) press or by doing a shift-click with mouse button 3 over the widget.

## The Canvas Widget

TKproE provides a configuration window to add, modify and delete items on a TCL/TK canvas. A sample screen shot of the window follows.



**Figure:** TKproE canvas configuration dialog

The *TKproE canvas configuration* dialog contains two expandable sections containing properties that are automatically displayed based on the type of canvas object being configured. Pressing mouse button 3 on some of these properties will display a special drop-down menu or dialog that allows an

# TKproE 2.20 Documentation

interactive selection of possible values. New values for properties take effect as soon as either:

- They are selected from a drop-down menu or special dialog
- The Enter key is pressed. Remember, this means that closing the properties dialog before pressing enter will not save changes to the property with the current focus.
- Focus is moved to another property

## Selecting the current canvas object

A canvas object must first be selected in order to modify its properties. The large list box on the right hand side of the dialog displays the tag names of each object on the canvas. This list may be sorted by either identification number (ID), assigned tag or object type. The current object is established by selecting the desired object from the list using mouse button one. The object's properties will then be displayed on the left side of the dialog. Once selected, a canvas object may also be deleted or assigned an event binding. Changes to the object properties will only affect the currently selected object.

## Adding a canvas object

Adding a new object consists of selecting an object type from the **Type** drop-down menu, entering its properties then pressing the **Insert** button.

New window objects for a canvas are most easily added using the Widget tool bar with the current widget path set to the desired canvas . Using the Widget tool bar will automatically create the canvas window object and attach the new widget to it. However, if you choose, you may add a window object separately then change the object -window property to specify some window.

## Specifying canvas object tags

The tags that should be attached to the current canvas object can be entered in the upper left hand side of the dialog.

## Adjusting canvas object coordinates

The position of an object can be adjusted using various controls in the **Coord** section of the dialog. In this section, the coordinates of the current object are displayed in a scrollable text box.

## *Moving an object*

The currently selected object may be moved one pixel at a time by pressing any of the four arrow buttons. Alternatively, the coordinates may be directly edited within the text box. In this case, the new coordinates will take effect when the Accept button (found amidst the arrow buttons) is pressed.

## *New coordinates*

New coordinates for an object may be either typed into the text box or added by pointing with the mouse. To use the mouse, first check the Point box then click with mouse button one at the canvas location you wish to copy to the text box. When using the pointing method, the coordinates will be added into the text box at the location of the insertion cursor.

## Other canvas object properties

After an object has been selected, its remaining options may be modified using the expandable Standard and Specific options bars. Clicking on these two bars will expand and contract the associated option sections. The options displayed within these expandable sections may be modified

# TKproE 2.20 Documentation

by typing in the new value. Alternatively, the property's associated configuration pop-up window may be displayed by pressing mouse button three over the property's entry box.

## Using the draw tools

The **Tools** section contains several tools that simplify the most common tasks associated with shaping or arranging objects on a canvas. Additionally, the tools allow canvas objects to be moved in permanently or temporarily defined groups.

The **Paste** button may be used to restore the last group of objects that was Cut.

The following draw modes are active when their respective radio buttons are selected. A description of each draw mode follows.

### **Attach**

In Attach mode, mouse button one can be used to attach a canvas object to an arbitrary object grouping. The objects do not have to be exclusively bounded by a common rectangular region. They may have any spacial relationship to each other. After several canvas objects are attached together, they can be selected to move as a unit when ObjMove mode is active. When Attach mode is active, objects will be attached to either an existing complex object group or a new complex grouping will be created. This behavior is dependent of the first object selected after switching to Attach mode. If the first object selected is already part of a complex object, any selected object will be added to this same complex object. If the first object doesn't yet belong to a complex object group, any selected object will be added to a new complex grouping.

### **Copy**

In this mode, a group of canvas objects may be highlighted using mouse button one and then copied as a group. The group is selected in the same manner as the GrpMove option. A copy of the objects are created and placed over the originals. The new objects will remain highlighted by a yellow border. The new objects may then be moved by pressing and holding mouse button one, moving the objects to the desired location then releasing the button.

### **Cut**

In this mode, a group of canvas objects may be highlighted using mouse button one and then deleted. The group is selected in the same manner as the GrpMove option. Once the mouse button is released the objects are deleted. The objects may be restored by using the **Paste** button.

### **Detach**

In Detach mode, mouse button one can be used to detach a canvas object from a more complex object grouping. Simply select the canvas object with mouse button one while in this mode. The selected object will be removed from any complex object grouping.

### **GrpMove**

In this mode, a group of canvas objects may be highlighted using mouse button one and then moved as a group. Mouse button one is first used to enclose the group of object by stretching a yellow border around them. The border is defined by pressing and holding mouse button one while moving the mouse to enclose the objects. Releasing the mouse button causes the yellow border to snap back to surround only those objects completely enclosed by the border. The enclosed objects may then be moved by pressing and holding mouse button one, moving the objects to the desired location then releasing the button.

# TKproE 2.20 Documentation

## **ID**

This mode retrieves the item ID number for a canvas object. Pressing mouse button one over an object while in ID mode displays the ID number at the top of the **Tools** section. This action will also select the item as the current canvas object.

## **Lower**

In this mode, press mouse button one over any canvas object you wish to place at the bottom of the object stacking order.

## **ObjMove**

ObjMove mode allows the user to move an arbitrary set of objects as a group. Canvas objects are grouped together using the Attach mode. Press and hold mouse button one over a member of the object group, move the mouse to the desired location then release the mouse button. Objects in the group maintain their spatial relationship to each other.

## **OneMove**

This mode is used to move a single object at a time. Simply select the object with mouse button one, hold down the button and move the object to the desired location then release the mouse button.

## **Raise**

In this mode, press mouse button one over any canvas object you wish to raise to the top of the object stacking order.

## **Stretch**

Any vertex of any object may be moved in this draw mode. The object dynamically reshapes as its vertex is moved.

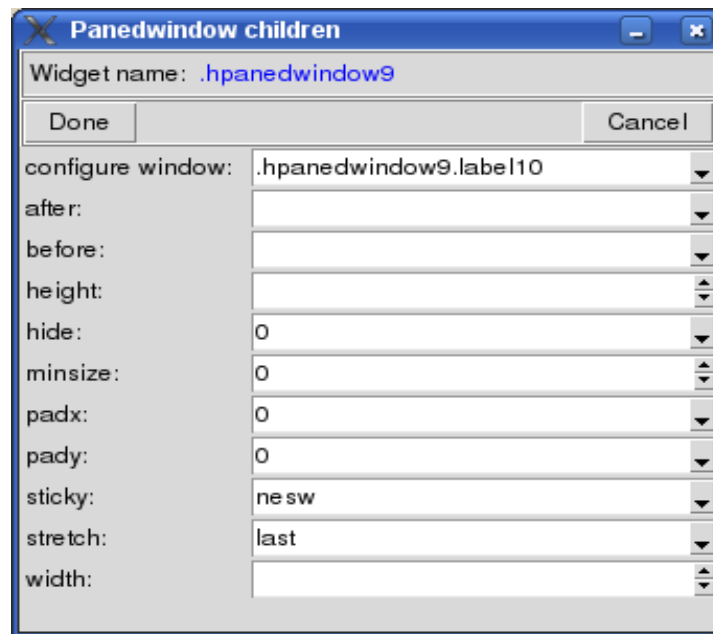
## **None**

Select this item to turn off all draw modes.

# TKproE 2.20 Documentation

## *The Paned Window Widget*

In addition to the standard widget properties dialog box, TKproE provides a dialog box to manage the geometry of items within the paned window. A sample screen shot of the dialog box follows.



**Figure:** TKproE paned window children dialog

This dialog box can be displayed by pressing the **More** button in the upper right corner of the standard properties dialog.

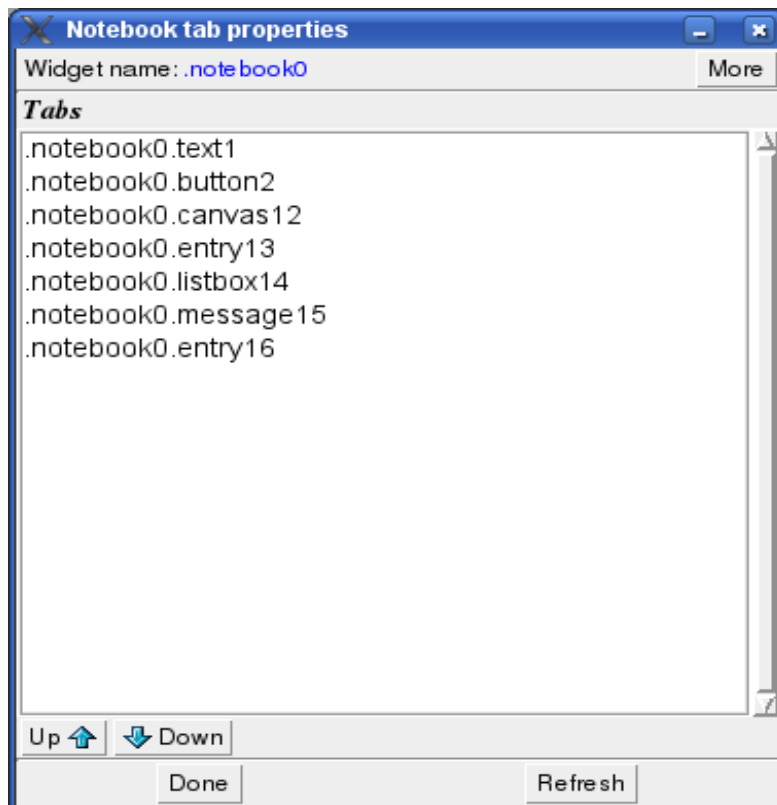
Use the combo-box associated with the “configure window” property to select the paned window child to configure.



# TKproE 2.20 Documentation

## The Notebook Widget

A special dialog displays when attempting to configure the properties of a notebook widget. This dialog box is used for managing the geometry of items within the notebook widget. A sample screen shot of the dialog box follows.



**Figure:** TKproE notebook tab properties dialog

To display the standard properties dialog box press the **More** button in the upper right corner of the “Notebook tab properties” dialog.

To refresh the list of managed tabs, press the **Refresh** button if tabs have been added or deleted from the notebook.

Press the **Done** button to close the dialog box.

Select the name of one of the tab windows in order to configure the tab. This will display the properties that can be configured for the tab as shown below.



## TKproE 2.20 Documentation

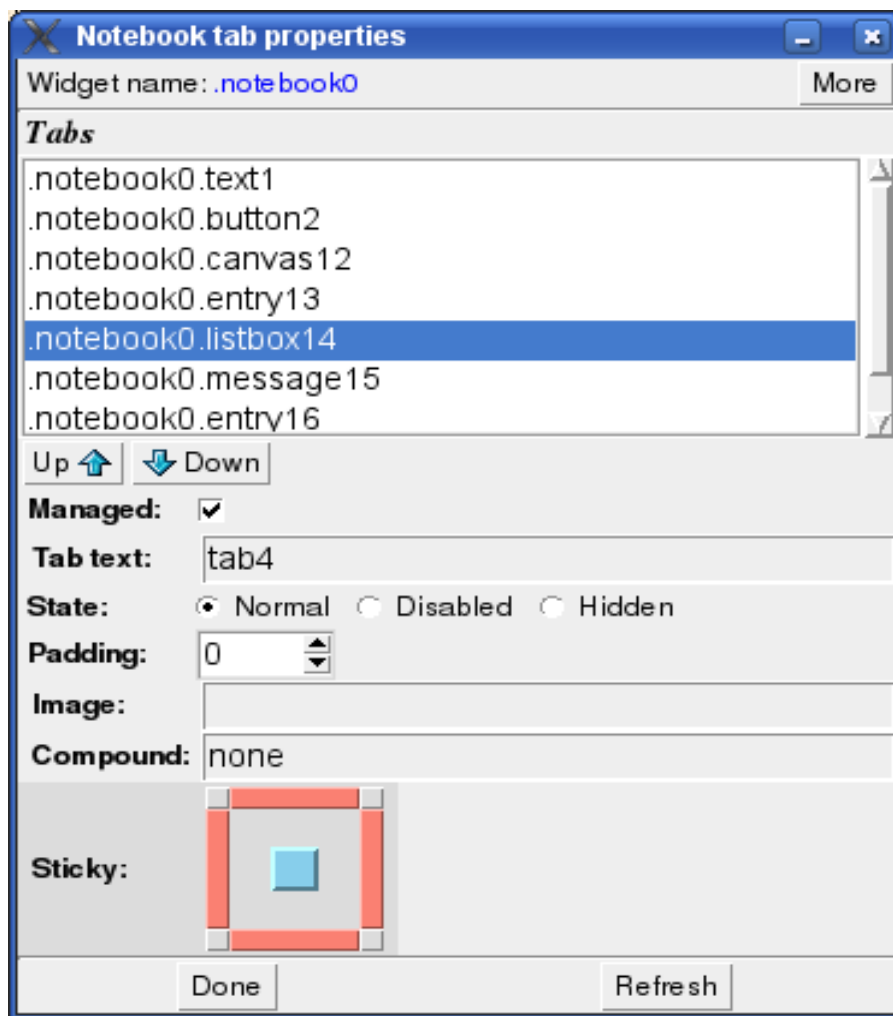


Figure: TKproE notebook tab properties with selected tab


Uncheck the **Managed** property to hide the tab.

The tab label can be specified along with an image to display on the selected tab. The **Compound** property controls whether and how the tab text, image or both are displayed

The child window's **sticky** property can be set using a set of eight buttons arranged in a rectangular border layout. Within this border is a small box used to simulate the behavior of the widget being configured. Pressing one of the eight buttons will cause the nearest side of the simulated widget to stick to the button. The selected button will also turn red. If a red button is pressed again, it will return to its original color and the simulated widget will no longer stick to the button. The child window of the tab will also reflect the changes as the sticky property is manipulated.

# TKproE 2.20 Documentation

## Menu configuration

TKproE provides a special dialog to interactively configure TK menus. This dialog can be displayed while editing the menu property of a menu button. In this case, simply, press mouse button three with the mouse cursor over the property's entry box. The **TKproE menu editor** may also be displayed by pressing the properties  action button with the current widget path selected to a menu.

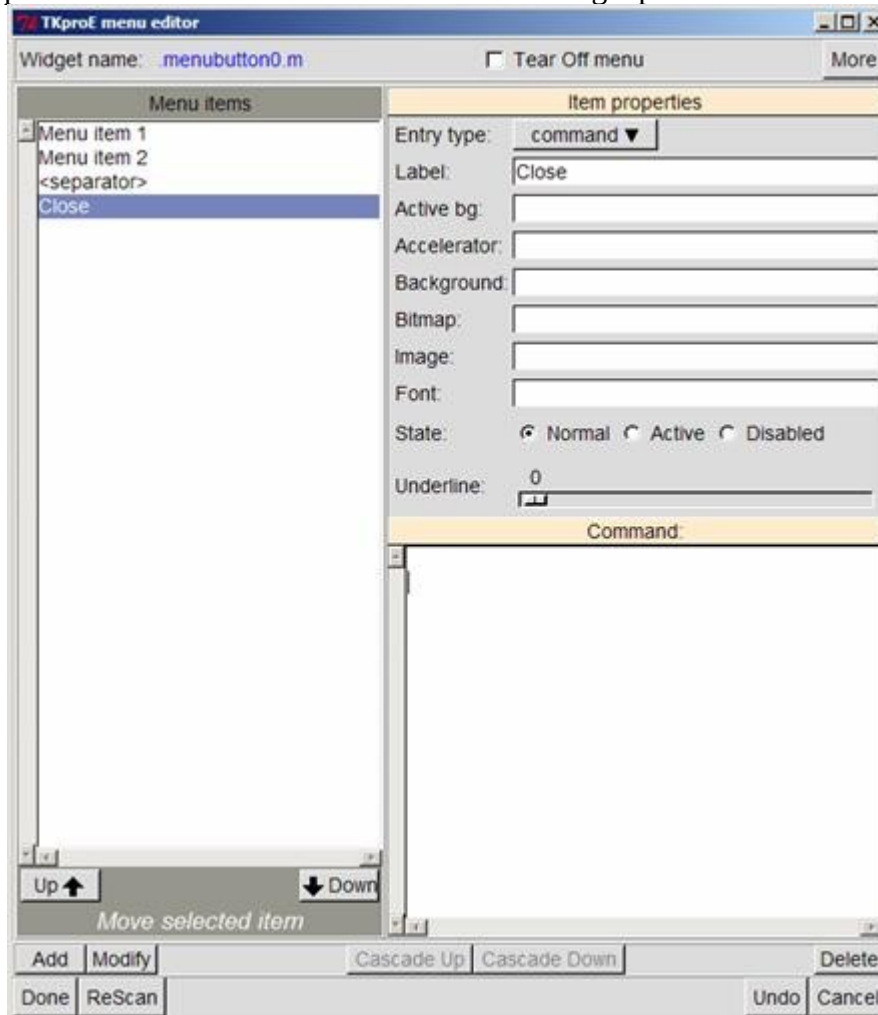


Figure: TKproE menu editor

The menu's entries are listed on the left hand side of the dialog. Selecting an item on the list will display its properties in the right hand side of the dialog. Once selected, the item properties may be modified. The changes will take effect once the **Modify** button is pressed.

Use the **Cascade Up** button to move (when possible) one level up in the menu hierarchy. Use the **Cascade Down** button to move one level down in the menu hierarchy when an item of type *Menu* is selected. These two buttons are only enabled while such movement is possible.

Pressing the **Add** button will insert a new menu item with the currently specified properties.

Pressing the **Delete** button will remove the selected item from the menu.

The **ReScan** button should be pressed to update the menu item list on the left if the menu has been modified by some other means since the TKproE menu editor was opened.

The **Undo** button can be used to undo any changes made to the menu since the TKproE menu editor

# TKproE 2.20 Documentation

was opened.

The **Cancel** button will perform the **Undo** button's actions then immediately close the dialog box.

The **Done** button will immediately close the dialog box.

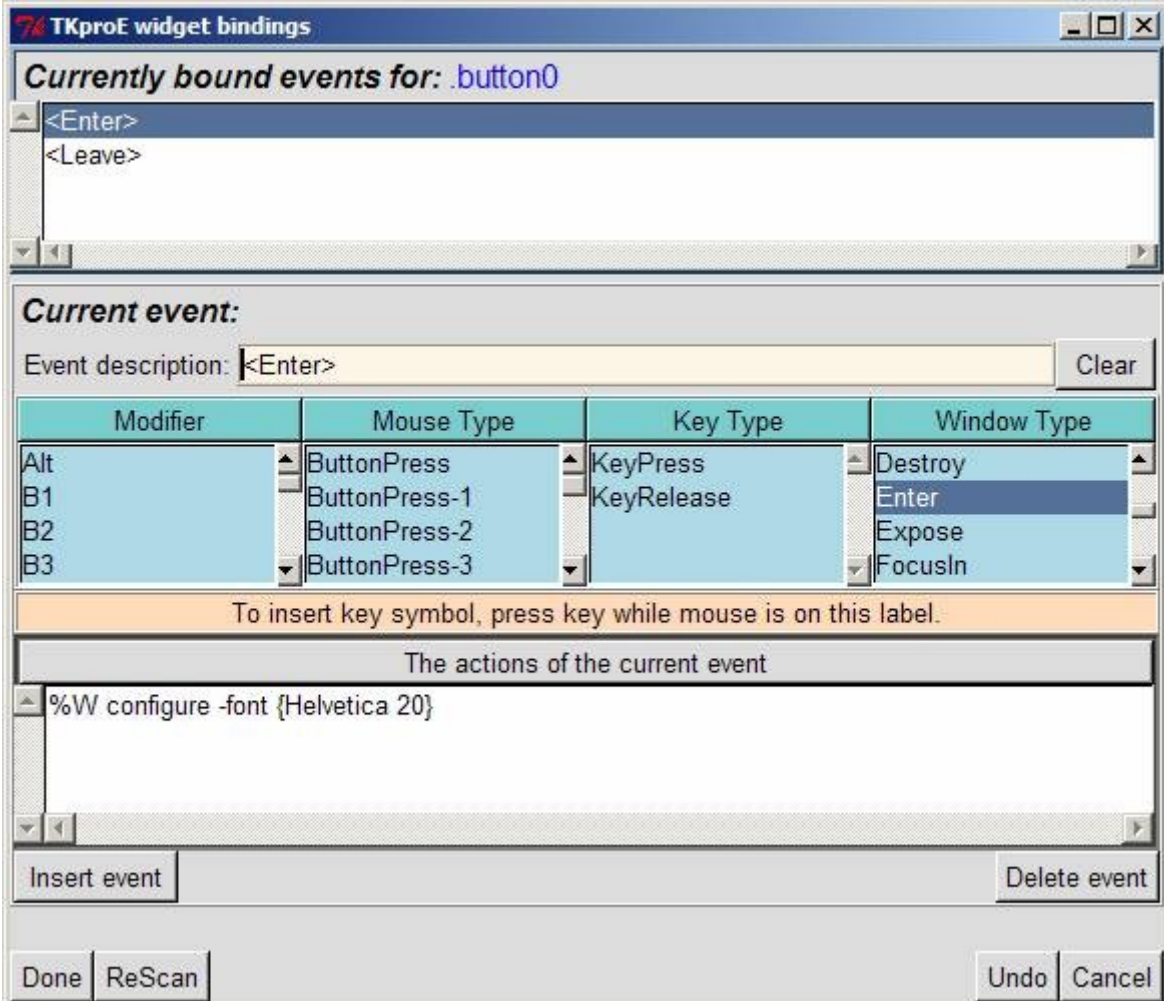
Pressing the **More** button will display the general Widget properties dialog for the menu.

## Widget bindings

Bindings can be configured using the dialog shown below. This dialog is displayed by pressing the



button on the [Action Button Bar](#).



The dialog box is titled "TKproE widget bindings". It has a header section "Currently bound events for: .button0" containing a listbox with "<Enter>" and "<Leave>". Below this is a section "Current event:" with an "Event description:" text field containing "<Enter>" and a "Clear" button. Underneath is a table with four columns: "Modifier", "Mouse Type", "Key Type", and "Window Type". The "Modifier" column lists "Alt", "B1", "B2", and "B3". The "Mouse Type" column lists "ButtonPress", "ButtonPress-1", "ButtonPress-2", and "ButtonPress-3". The "Key Type" column lists "KeyPress" and "KeyRelease". The "Window Type" column lists "Destroy", "Enter", "Expose", and "FocusIn", with "Enter" currently selected. Below the table is an instruction: "To insert key symbol, press key while mouse is on this label." This is followed by a section "The actions of the current event" containing a text area with the code "%W configure -font {Helvetica 20}". At the bottom are buttons for "Insert event", "Delete event", "Done", "ReScan", "Undo", and "Cancel".

Modifier	Mouse Type	Key Type	Window Type
Alt	ButtonPress	KeyPress	Destroy
B1	ButtonPress-1	KeyRelease	Enter
B2	ButtonPress-2		Expose
B3	ButtonPress-3		FocusIn

This form can be used to specify the TCL code that should be executed when a specific event occurs within a widget. First, select the event from the four list boxes shown. Second, enter TCL code in the text box for the actions of the current event. Finally, press the **Insert event** button to create the binding. Once created, the bound event will be shown in the listbox at the top of the dialog. This listbox will contain a list of all existing bindings for the currently selected widget. Therefore, it may already list some bindings when first opened.

# TKproE 2.20 Documentation

An existing binding can be edited by selecting it in the listbox at the top of the dialog. This will display the event's description and associated TCL code in other parts of the dialog. Simply edit the associated code and press the **Insert event** button when done.

Pressing the **Clear** button will erase the event description text. Doing this will allow the user to easily specify a completely new event.

An existing binding can be deleted by selecting it in the listbox at the top of the dialog then pressing the **Delete event** button.

Press the ReScan button to update the list of bindings without the need to close then re-open the dialog. This would only be necessary if some part of the application under development dynamically changed the bindings and you wished to reflect the changes in the dialog's binding list.

All binding editing for the dialog's session can be undone by pressing the **Undo** button.

Pressing the **Cancel** button will both undo the changes and close out the dialog.

Press the **Done** button to close the dialog.


## Geometry Management

When you insert new widgets into your application with TKproE, the placement of the widget will be controlled using the currently selected TK geometry manager (the *packer*, the *placer* or the *gridder*). The default geometry manager may be selected on the TKproE general options dialog. This pop-up window can be displayed by selecting the General item on the **Options** menu.

It is possible to combine widget geometry management methods in the same toplevel window. However, since geometry managers ignore the widgets managed with other geometry managers, care is needed to prevent widgets from accidentally overlapping. The combination of multiple methods should be used only by the experienced users who understand the geometry manager behavior well.

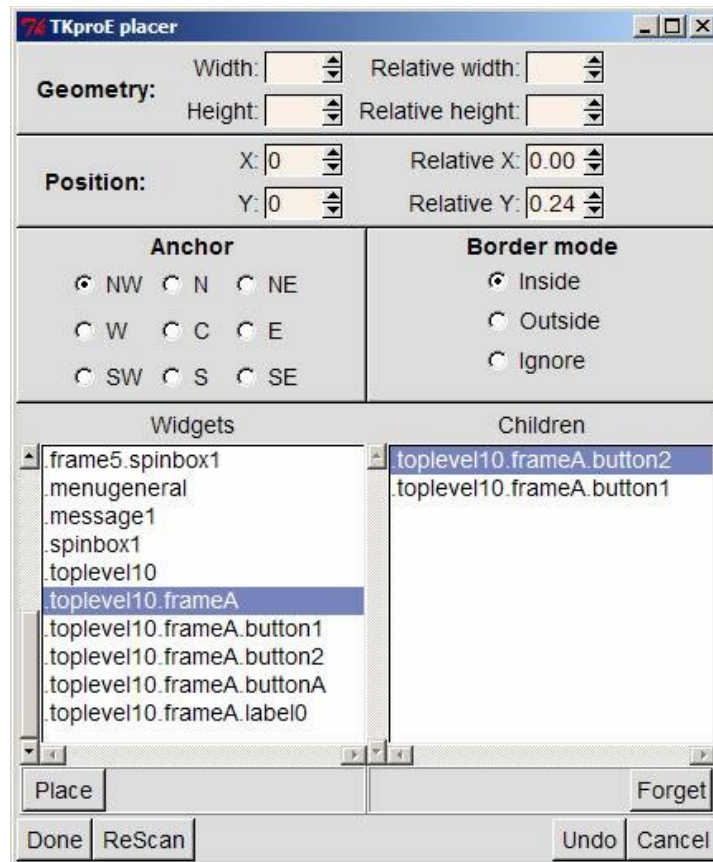
A separate dialog box is provided for each of the TK geometry managers. They are described below and provide a means for setting all options associated with that geometry manager.

### The Placer

The TKproE placer dialog can be displayed by pressing the  action button. The displayed dialog box will allow you to set all placer geometry properties of the [currently selected widget](#).

Using the TKproE placer dialog, widgets can be selected, placed and rearranged. See the TK documentation included with the TCL/TK scripting language for detailed explanations of the placer options.

# TKproE 2.20 Documentation



**Figure:** TKproE placer dialog

With this dialog box, it is possible to change the placing of the complete widget tree.

The top area contains the available placer options. The options show the current setting of the child selected in the right list at the bottom. Placer geometry settings that are changed using the dialog box will take effect immediately.

Selecting another child from the list on the right will update the settings. The left list contains the widget tree. By double clicking at a widget in that list, this widget is made the current master. The right list will then update to contain the children placed by this master.

Since there are multiple independent geometry managers, at some time you may wish to change how a child of a widget should be managed. This is done with the two buttons **Place** and **Forget**.

Pressing the **Forget** button will remove the widget currently selected in the right list from management by the placer.

Pressing the **Place** button will cause the placer to manage the widget currently selected in the left list.

The **ReScan** button should be pressed to update the widget tree list on the left if widgets have been added or deleted since the TKproE Placer dialog was opened.

The **Undo** button can be used to undo any changes made to the currently selected widget since it was selected.

The **Cancel** button will perform the **Undo** button's actions then immediately close the dialog box.

The **Done** button will immediately close the dialog box.

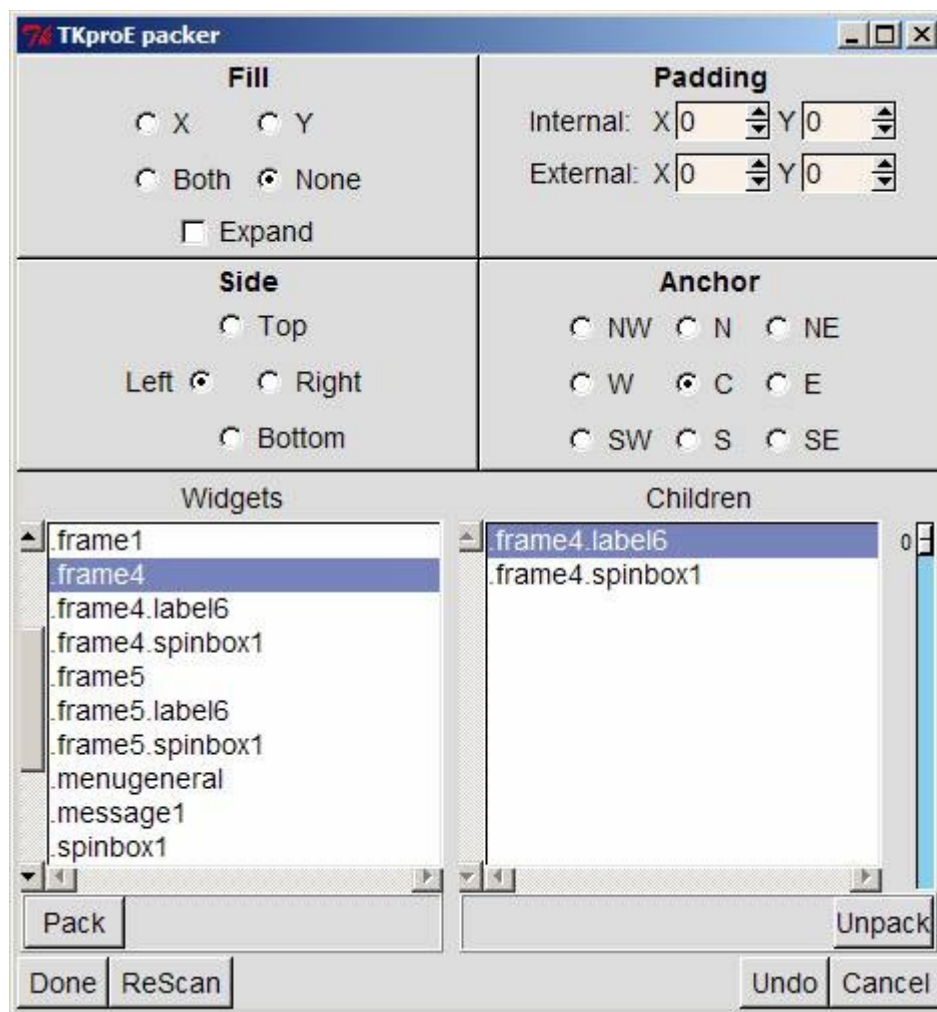
# TKproE 2.20 Documentation

## The Packer

The TKproE packer dialog can be displayed by pressing the  action button. The displayed dialog box will allow you to set all packer geometry properties of the [currently selected widget](#).

When managing widgets with the packer, it is important to group widgets with frame widgets. The user can use the frames as row-column widgets. All children of one frame are usually packed to one side of the parent.

Using the TKproE packer dialog, widgets can be selected, packed and rearranged. See the TK documentation included with the TCL/TK scripting language for detailed explanations of the packer options.



**Figure:** TKproE packer dialog

With this dialog box, it is possible to change the packing of the complete widget tree.

The top area contains the available packer options. The options show the current setting of the child selected in the right list at the bottom. Packer geometry settings that are changed using the dialog box will take effect immediately.

Selecting another child from the list on the right will update the settings. The left list contains the widget tree. By double clicking at a widget in that list, this widget is made the current master. The right list will then update to contain the children packed by this master.

# TKproE 2.20 Documentation

Since there are multiple independent geometry managers, at some time you may wish to change how a child of a widget should be managed. This is done with the two buttons **Pack** and **Unpack**.

Pressing the **Unpack** button will remove the widget currently selected in the right list from management by the packer.

Pressing the **Pack** button will cause the packer to manage the widget currently selected in the left list.


The **ReScan** button should be pressed to update the widget tree list on the left if widgets have been added or deleted since the TKproE Packer dialog was opened.

The **Undo** button can be used to undo any changes made to the currently selected widget since it was selected.

The **Cancel** button will perform the **Undo** button's actions then immediately close the dialog box.

The **Done** button will immediately close the dialog box.

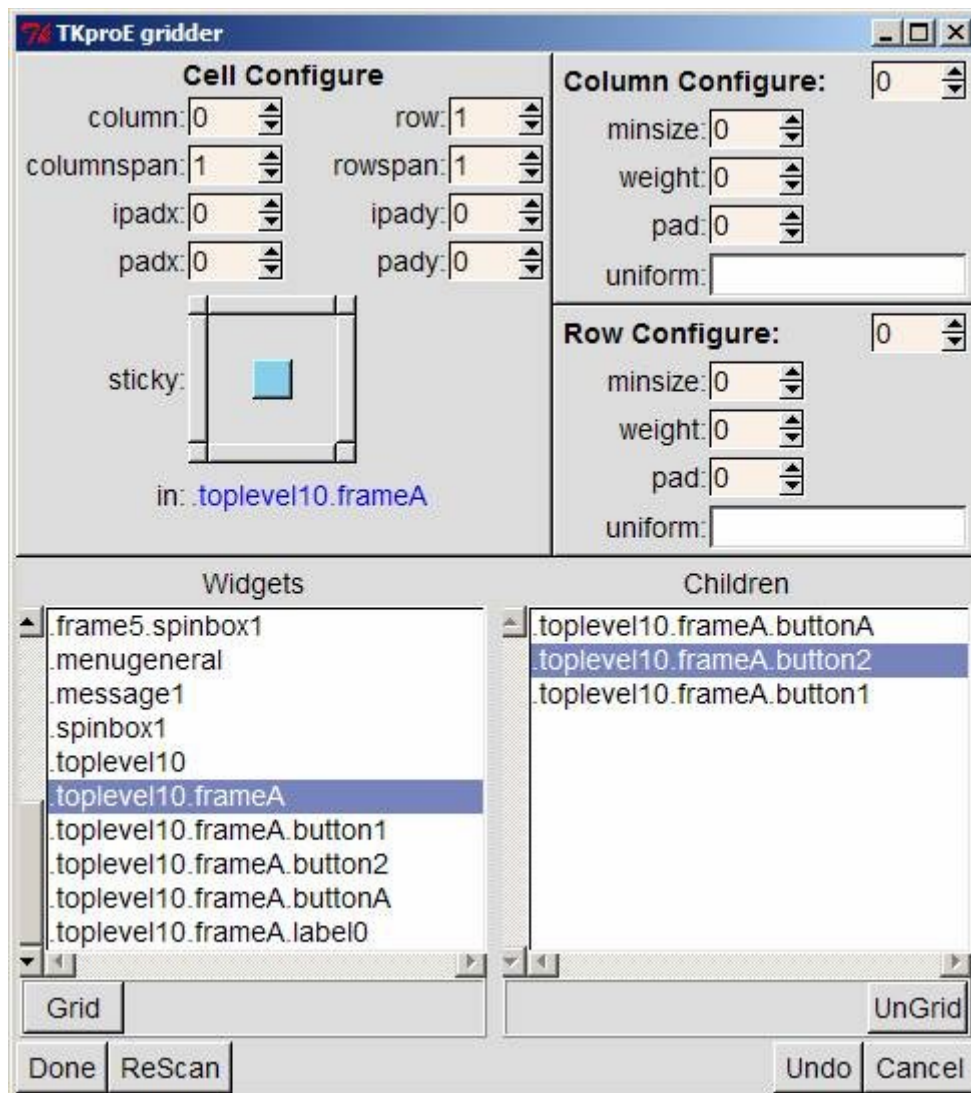
## ***The Gridder***

The TKproE gridder dialog can be displayed by pressing the  action button. The displayed dialog box will allow you to set all gridder geometry properties of the [currently selected widget](#).

Using the TKproE gridder dialog, widgets can be selected, gridded and rearranged. See the TK documentation included with the TCL/TK scripting language for detailed explanations of the gridder options.



# TKproE 2.20 Documentation



**Figure:** TKproE gridder dialog

With this dialog box, it is possible to change the gridding of the complete widget tree.

The top area contains the available gridder options. The options show the current setting of the child selected in the right list at the bottom. Gridder geometry settings that are changed using the dialog box will take effect immediately.

Selecting another child from the list on the right will update the settings. The left list contains the widget tree. By double clicking at a widget in that list, this widget is made the current master. The right list will then update to contain the children packed by this master.

The cell **sticky** property can be set using a set of eight buttons arranged in a rectangular border layout. Within this border is a small box used to simulate the behavior of the widget being configured. Pressing one of the eight buttons will cause the nearest side of the simulated widget to stick to the button. The selected button will also turn red. If a red button is pressed again, it will return to its original color and the simulated widget will no longer stick to the button. The widget under configuration will also reflect the changes as the sticky property is manipulated.

Since there are multiple independent geometry managers, at some time you may wish to change how



# TKproE 2.20 Documentation

a child of a widget should be managed. This is done with the two buttons **Grid** and **UnGrid**.

Pressing the **UnGrid** button will remove the widget currently selected in the right list from management by the gridder.

Pressing the **Grid** button will cause the gridder to manage the widget currently selected in the left list.

The **ReScan** button should be pressed to update the widget tree list on the left if widgets have been added or deleted since the TKproE Packer dialog was opened.

The **Undo** button can be used to undo any changes made to the currently selected widget since it was selected.

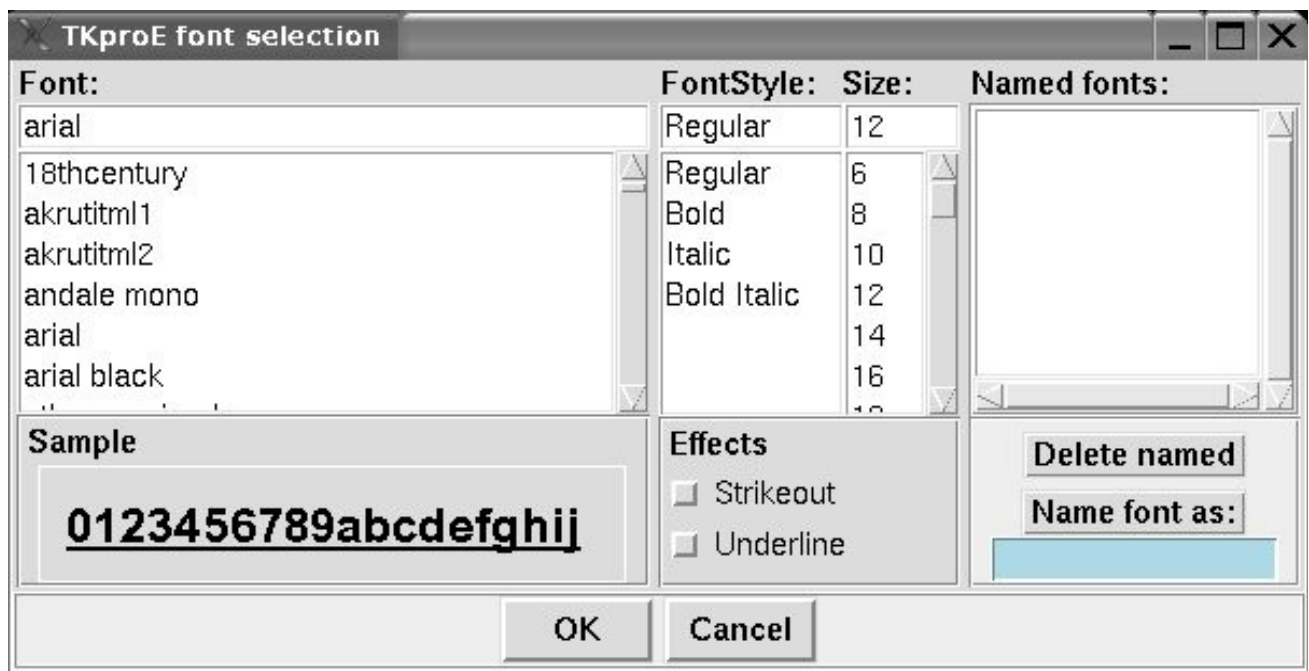
The **Cancel** button will perform the **Undo** button's actions then immediately close the dialog box.

The **Done** button will immediately close the dialog box.

## Font Selection and Management

TKproE provides the dialog box shown below for performing the following tasks.

- Choosing the individual font characteristics of a TK widget
- Assigning a named font to a widget
- Create a new TK named font
- Delete an existing TK named font
- Modify an existing TK named font
- Viewing a sample of the selected font



A list of available font families is displayed on the left side of the dialog box. A list of named fonts found in your application is shown on the right side of the dialog box.

Selecting a named font from the list will set the currently selected font characteristics to those of the

# TKproE 2.20 Documentation

named font.

A named font can be removed from the list by first selecting it with the mouse then pressing the **Delete named** button. Remember, however, the deletion will be successful only if the named font is no longer associated with any widget in the application.

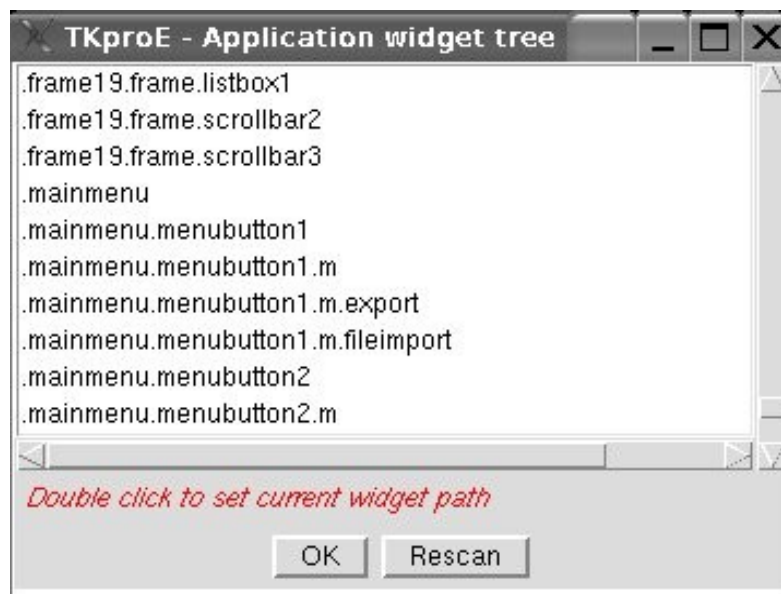
Entering a name below the **Name font as:** button and then pressing the button will create a new font by that name. The new font will have the currently selected characteristics displayed in the dialog box.

When this dialog is used to set the font property of a widget, pressing the **OK** button will close the dialog box and set the font properties to the selected values.

Pressing the **Cancel** button will simply close the dialog box.

## Application widget tree

The dialog box shown below displays a list of all TK objects (e.g., buttons, toplevel windows) currently displayed by your application. This is one of the dialog boxes that can be displayed from the Windows option of the [TKproE menu bar](#).



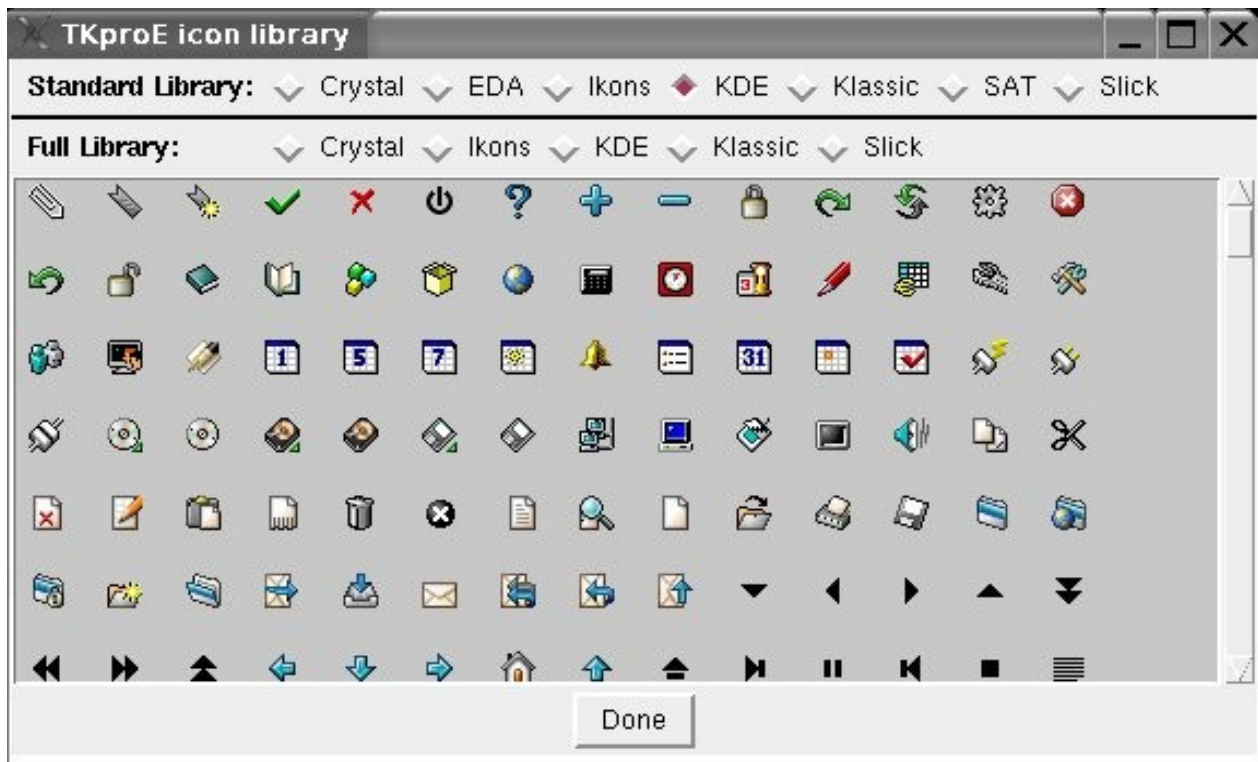
Double-clicking on an item in the list will set the current widget path to the selected item.

The **ReScan** button can be pressed to update the widget tree list if widgets have been added or deleted since the dialog box was opened.

Press the **OK** button to close the dialog box.

# TKproE 2.20 Documentation

## Icon Library



The TKproE icon library dialog box, shown above, provides access to an extensive library of icons that can be quickly incorporated into applications. These icon images were collected from the ICONS package web site by Adrian Davis (<http://www.satisoft.com/tcltk/icons/index.html>) and from the Silk Icons web site by Mark James (<http://www.famfamfam.com/lab/icons/silk/>).

These icons are stored in the *lib/icons* directory of the TKproE distribution.

The original sources for the icon packages are listed in the next table.

<b>Icon package</b>	<b>Package Create/Maintainer</b>
KDE	KDE development team
Crystal	Everaldo
iKons	Kristof Borrey
Klassic	Asif Ali Rizwaan
Slick	Rob Cosgrove
EDA	Shaun Deacon
Silk	Mark James

All standard libraries contain equivalent icons (but with different themes), and are therefore interchangeable. Where necessary, icons from another source were used to fill in where equivalent icons are not available.

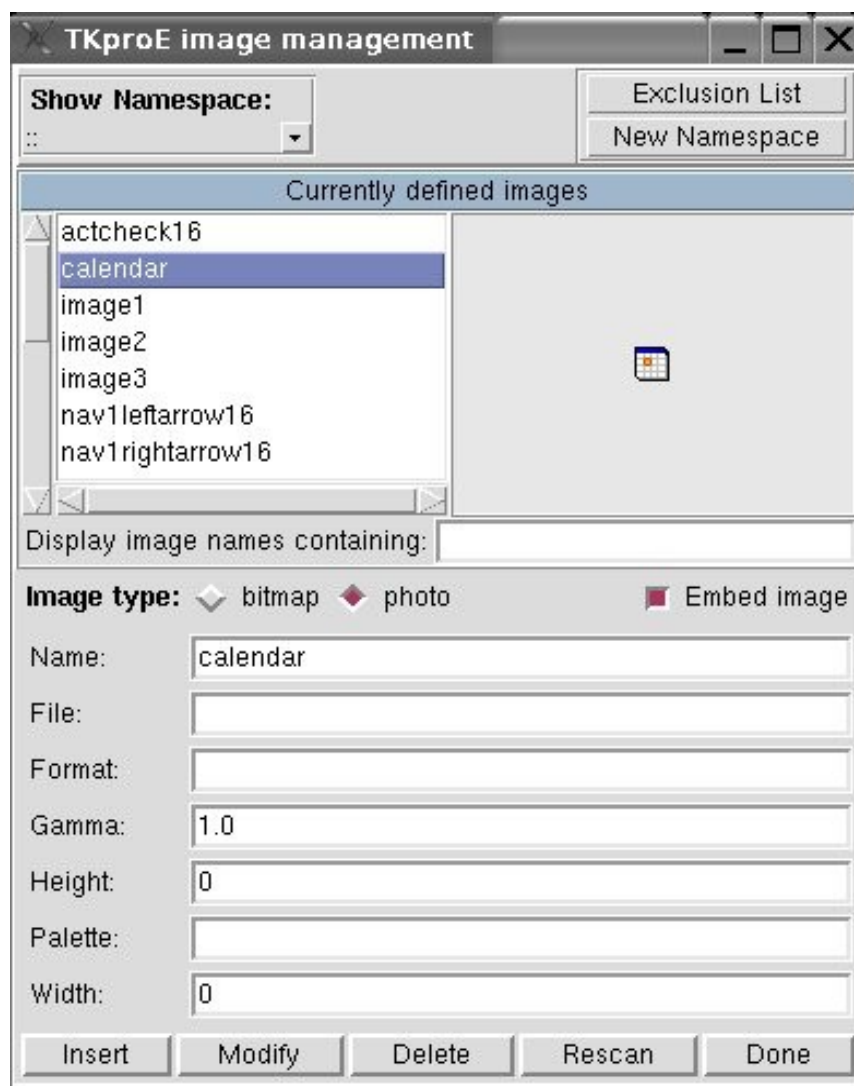
# TKproE 2.20 Documentation

The full libraries contain the complete set of icons available from the creator/maintainer. They vary in the number and names of the icons they contain.

Selecting any one of the standard or full libraries will display their associated icons in the dialog's scrollable window. Selecting one of these icons with mouse button 1 will display the [TKproE image management dialog box](#) which will be properly filled in for the creation of a new TK image based on the icon. You can then change the image name to one that you prefer and press the **Insert** button to create the new image.

Press the **Done** button on the icon library dialog to close it.

## Image Management



The dialog box shown above is used to manage your TK applications images. TK images can be created, deleted and configured using this dialog box. This dialog is displayed by selecting the Windows/Images option on the [TKproE Menu Bar](#).

Images are administered separately for each Namespace known to the user application. The current

# TKproE 2.20 Documentation

Namespace is selected using a drop-down menu in the upper left section of the dialog.

TKproE makes some Namespaces (e.g., `::tcl` and `::tk`) read-only. These Namespaces are placed on an exclusion list that can be displayed using the **Exclusion List** button in the upper right hand side of the dialog. Additional Namespaces may be added to the exclusion list by the user.

A new Namespace can be easily created by pressing the **New Namespace** button

Names of existing images are displayed in the listbox in the upper left of the dialog box. Selecting a name in this listbox will display the image to the right of the list box. The properties for the image can be edited then re-saved by pressing the **Modify** button.

A new image can be added by entering its desired name, selecting the image type, entering the file system path to the image and then pressing the **Insert** button.

If the embed image option is selected the image data will be embedded directly into the application script when it is saved by TKproE. The original image files for such images do not need to be distributed with the application. The image data is included into the TCL script as base-64 encoded text.

Press the **Rename** button to change the name of the currently selected variable.

Press the **Delete** button to delete the currently selected image.

The list of existing images can be updated by pressing the **Rescan** button. This would be necessary if you created new images by issuing commands in the WISH console window after the dialog was opened.

Press the **Done** button to close the dialog box.

## Additional Widget Support

TKproE provides support for some widgets beyond those found in standard TK. These additional widgets can be added to an application by typing their appropriate command in the WISH console or by inserting a [template](#) (included in the TKproE distribution) for the new widget. The following table lists the additional widgets currently supported.

Widget type	Description	Template file name
Table	This is the Tktable widget by Jeffrey Hobbs.	Tktable.tcl
BLT graph	This is the BLT graph widget by George Howlett	BLTgraph.tcl

# TKproE 2.20 Documentation

## Advanced Features

### Templates

With TKproE a user can save complex widget structures into files called templates. These files contain TCL/TK code defining a widget. A template can be inserted by the user, adding this widget structure to the application program at the current widget path. Some templates are included with the TKproE file distribution.

By default, TKproE saves template files into the *templates* directory immediately below the TKproE installation directory. However, any directory may be chosen when saving the template. To save a template the user must:

1. Cut or copy the current widget.
2. Use the **Edit/Save cutbuffer** option of the TKproE menu bar to open the *File Save* pop-up window.
3. Select the subdirectory to save the template to.
4. Specify the file name of the template.
5. Complete the process by pressing the **Save** button on the *File Save* pop-up.

### Source modules and Namespaces

A TKproE generated program is saved into one file, containing the complete code. Namespaces loaded into your application are automatically saved unless one of the following applies:

1. The namespace is one of:
  - a. ::auto\_mkindex\_parser
  - b. ::freewrap
  - c. ::msgcat
  - d. ::pkg
  - e. ::tcl
  - f. ::tk
  - g. ::tk::dialog
  - h. ::tk::dialog::error
  - i. ::tk::msgcat
  - j. ::tk::panedwindow
  - k. ::tk::spinbox
  - l. ::zvfs
2. You have placed the namespace on the TKproE Namespace Exclusion List dialog box.

During development, you may wish to use namespaces to separate different portions of the application. Individual namespaces may be saved by using the **File/Save Namespace** option on the menu bar. This option allows saving only the variables, images and procedures assigned to that namespace.

# TKproE 2.20 Documentation

## The generated code

Source code generated by TKproE has a specific layout, and contains a certain functionality. The generated code is saved into one file.

In addition to the procedures that are written by the user, TKproE saves a number of automatically generated procedures that are used to create the widget structures and initialize the program. The source code is saved to file in the following order.

### ***TPstartupSrc and TPendSrc procedures***

These two procedures contain source that is evaluated during the startup of the application program. **TPstartupSrc** is executed as the first code in the application, and **TPendSrc** is executed after all toplevel windows have been displayed. The user should add initialization code inside one of these procedures.

### ***Image definitions***

This part of the code contains the necessary commands to replicate the images that existed when the application was saved. Images that have been defined with TKproE's **Embed image** option will have their image data embedded directly in this part of the code. The original image files for such images do not need to be distributed with the application. The image data is included into the TCL script as base-64 encoded text.

### ***Named fonts***

This part of the code contains the necessary commands to generate the named fonts that existed when the application was saved. The standard named fonts that are part of TK will not be saved. However, any named fonts created by the developer will be.

### ***Variable initialization procedure***

The user application's global variables are re-created and initialized with the procedure **TPinitVars**. The values assigned in the **TPinitVars** procedure represent the values when the application was saved.

Although the **TPinitVars** procedure restores the global variables to the exact value they had at the time the application was saved within TKproE, the developer should initialize important variables directly in the application's own initialization code.

### ***Toplevel creation procedures***

The procedures **TPopenWindow...**, **TPcloseWindow...**, etc. are created automatically by TKproE, to display and hide the various toplevel windows of the application. For example, the procedure **TPopenWindow.about** would be used to display the *.about* toplevel window. Likewise, the procedure **TPcloseWindow.about** would be used to close the *.about* toplevel window.

When a **TPcloseWindow** procedure is called it causes TKproE to update the associated **TPopenWindow** procedure to reflect any changes (e.g., newly added buttons) to the content of the toplevel window. Therefore, when coding an application, only the associated **TPcloseWindow** procedure should be used to close the window. This will guarantee the automatic saving of changes made while developing under TKproE.

There may also be procedures named **TPstartupSrc...** and **TPendSrc...**. They are called by the

# TKproE 2.20 Documentation

corresponding **TPopenWindow** procedure, and can be used to initialize the widgets that are inside the displayed toplevel window. **TPstartupSrc** and **TPendSrc** procedures can be created and edited using the *TKproE toplevel window properties* dialog.

## ***User defined procedures***

The procedures that have been written by the developer are all stored in this section.

## ***Invocation of TPstartupSrc***

The startup code is evaluated by running the procedure **TPstartupSrc**.

## ***Invocation of TPinitVars procedure***

The global variables are re-created by running the procedure **TPinitVars**.

## ***Display the toplevel windows***

The toplevel windows that should be displayed are created by invoking the appropriate **TPopenWindow...** procedures.

## ***Invocation of TPendSrc procedure***

The last code to be run prior to the application entering TCL/TK's event loop is evaluated by running the procedure **TPendSrc**.



# TKproE 2.20 Documentation

## Converting existing TCL/TK applications to TKproE

Since TKproE automatically generates application framework code for your program there are a few steps that should be done to transition an existing TCL/TK application to a TKproE based project.

Perform the following steps for a successful transition.

1. Start TKproE.
2. Load your application source code using TKproE's **File/Open** menu option. Display each of your application's toplevel windows.
3. Press the [Current widget path button on the Main TKproE dialog](#). This will display a pop-up window containing a list of all toplevel windows currently defined for the application under development. This will also cause TKproE to generate its own [procedures for opening and closing the application windows](#). Opening this Toplevel Windows dialog box causes TKproE to perform the following:
  1. Create a **TPopenWindow** procedure for each of the application's opened toplevel windows. For example: a procedure named **TPopenWindow.about** will be created for the .about toplevel window.
  2. Create a **TPcloseWindow** procedure for each of the application's opened toplevel windows. For example: a procedure named **TPcloseWindow.about** will be created for the .about toplevel window.
4. Close the *Toplevel windows* pop-up displayed by the previous step.
5. Modify the toplevel windows so that they will be closed using the newly generated **TPcloseWindow** procedures.
6. Modify each toplevel window's TPstartupSrc and TPendSrc [special procedures](#) to include any necessary initialization code.
7. The procedures originally used by the application to open and close its toplevel windows are no longer needed. They may now be deleted using the [TKproE Procedures dialog](#).
8. Create a procedure that can be used to set the global variables to the values desired when first opening the application. TKproE creates a snapshot of the application state when saving the project. This means that all global variables will be restored to the values they had at the time the application was saved. Therefore, it will probably be necessary to create a procedure that can be used to set the variables to the values desired when first opening the application. This procedure can then be called within the [TPendSrc](#) procedure.
9. Use the TKproE **File/Save As** menu option to save the converted application.