NAME

tixwish − Windowing shell for interpreting Tix commands.

SYNOPSIS

tixwish ?*fileName arg arg ...*?

OPTIONS

−display *display* Display (and screen) on which to display window.

−geometry *geometry* Initial geometry to use for window. If this option is specified, its value is stored in the **geometry** global variable of the application's Tcl interpreter.

−name *name* Use *name* as the title to be displayed in the window, and as the name of the interpreter for **send** commands.

−sync Execute all X server commands synchronously, so that errors are reported immediately. This will result in much slower execution, but it is useful for debugging.

DESCRIPTION

Tixwish is a simple program consisting of the Tcl command language, the Tk toolkit, and a main program that reads commands from standard input or from a file. It creates a main window and then processes Tcl commands. If **tixwish** is invoked with no arguments, or with a first argument that starts with "−", then it reads Tcl commands interactively from standard input. It will continue processing commands until all windows have been deleted or until end-of-file is reached on standard input. If there exists a file **.tixwishrc** in the home directory of the user, **tixwish** evaluates the file as a Tcl script just before reading the first command from standard input.

If **tixwish** is invoked with an initial *fileName* argument, then *fileName* is treated as the name of a script file. **Tixwish** will evaluate the script in *fileName* (which presumably creates a user interface), then it will respond to events until all windows have been deleted. Commands will not be read from standard input. There is no automatic evaluation of **.tixwishrc** in this case, but the script file can always **source** it if desired.

OPTIONS

Tixwish automatically processes all of the command-line options described in the **OPTIONS** summary above. Any other command-line arguments besides these are passed through to the application using the **argc** and **argv** variables described later.

APPLICATION NAME AND CLASS

The name of the application, which is used for purposes such as **send** commands, is taken from the **−name** option, if it is specified; otherwise it is taken from *fileName*, if it is specified, or from the command name by which **tixwish** was invoked. In the last two cases, if the name contains a "/" character, then only the characters after the last slash are used as the application name.

The class of the application, which is used for purposes such as specifying options with a **RESOURCE_MANAGER** property or .Xdefaults file, is the same as its name except that the first letter is capitalized.

VARIABLES

Tixwish sets the following Tcl variables:

argc Contains a count of the number of *arg* arguments (0 if none), not including the options described above.

argv Contains a Tcl list whose elements are the *arg* arguments (not including the options described above), in order, or an empty string if there are no *arg* arguments.

argv0 Contains *fileName* if it was specified. Otherwise, contains the name by which **tixwish** was invoked.

geometry If the **−geometry** option is specified, **tixwish** copies its value into this variable. If the variable still exists after *fileName* has been evaluated, **tixwish** uses the value of the variable in a **wm geometry** command to set the main window's geometry.

tcl_interactive Contains 1 if **tixwish** is reading commands interactively (**fileName** was not specified and standard input is a terminal-like device), 0 otherwise.

X RESOURCES

Tixwish makes use of several X Resources to determine the **Toolkit Options** for the Tix library. These X resources must be set using **RESOURCE_MANAGER** properties or .Xdefaults files **before tixwish** starts running. These resources must be associated with the main window of the **tixwish** application. These options include:

Name: **tixScheme**
Class: **TixScheme**

Specifies the color scheme to use for the Tix application. Currently only these schemes are supported: Blue, Gray, SGIGray, TixGray, and TK.

Name: **tixFontSet**
Class: **TixFontSet**

Specifies the FontSet to use for the Tix application. A FontSet designates the fonts to use for different types of widgets. Currently only these FontSets are supported: 12Point, 14Point and TK.

For example, you may put these two lines in your .Xdefaults file

```
*tixwish.tixScheme: Gray
*tixwish.tixFontSet: 12Point
```

SCRIPT FILES

If you create a Tcl script in a file whose first line is

#!/usr/local/bin/tixwish

then you can invoke the script file directly from your shell if you mark it as executable. This assumes that **tixwish** has been installed in the default location in /usr/local/bin; if it's installed somewhere else then you'll have to modify the above line to match. Many UNIX systems do not allow the **#!** line to exceed about 30 characters in length, so be sure that the **tixwish** executable can be accessed with a short file name.

PROMPTS

When **tixwish** is invoked interactively it normally prompts for each command with "**%** ". You can change the prompt by setting the variables **tcl_prompt1** and **tcl_prompt2**. If variable **tcl_prompt1** exists then it must consist of a Tcl script to output a prompt; instead of outputting a prompt **tixwish** will evaluate the script in **tcl_prompt1**. The variable **tcl_prompt2** is used in a similar way when a newline is typed but the current command isn't yet complete; if **tcl_prompt2** isn't set then no prompt is output for incomplete commands.

KEYWORDS

shell, wish, Tk, toolkit

NAME

tixBalloon − Create and manipulate tixBalloon widgets

SYNOPSIS

tixBalloon *pathName ?options*?

SUPER-CLASS

The **tixBalloon** class is derived from the **TixShell** class and inherits all the commands, options and subwidgets of its super-class.

STANDARD OPTIONS

The Balloon widget supports all the standard options of a frame widget. See the **options(n)** manual entry for details on the standard options.

WIDGET-SPECIFIC OPTIONS

Name: **initWait**
Class: **InitWait**
Switch: **−initwait**

In milliseconds. Specifies how long the balloon should wait before popping up in a widget.

Name: **state**
Class: **State**
Switch: **−state**

Specifies the which help message to display when the mouse pointer enters a widget associated with this balloon. Valid options are **both**: display both the balloon message and the status bar message, **balloon**: display only the balloon message, **status**: display only the status bar message and **none**: display no messages.

Name: **statusBar**
Class: **statusBar**
Switch: **−statusbar**

Specifies the widget to use as the status bar of this balloon. This widget must have a "-text" option. Usually a label widget is used.

SUBWIDGETS

Name: **label**
Class: **Label**

The label widget that shows the little arrow bitmap in the pop-up balloon window.

Name: **message**
Class: **Label**

The message widget that shows the descriptive message in the the pop-up balloon window.

DESCRIPTION

The **tixBalloon** command creates a new window (given by the *pathName* argument) and makes it into a Balloon widget. Additional options, described above, may be specified on the command line or in the option database to configure aspects of the Balloon widget such as its cursor and relief.

The Balloon widget can be used to show popped-up messages that describe the functions of the widgets in an application. A Balloon widget can be bound to a number of widgets. When the user moves the cursor inside a widget to which a Balloon widget has been bound, a small pop-up window with a descriptive message will be shown on the screen.

WIDGET COMMANDS

The **tixBalloon** command creates a new Tcl command whose name is the same as the path name of the Balloon widget's window. This command may be used to invoke various operations on the widget. It has the following general form:

pathName option ?*arg arg ...*?

PathName is the name of the command, which is the same as the Balloon widget's path name. *Option* and the *arg*s determine the exact behavior of the command. The following commands are possible for Balloon widgets:

pathName **bind** widget ?*option value ...* ?

Binds the Balloon widget to the *widget*. The messages to be shown can be passed as extra arguments to this command in *option value* pairs. Possible options: **−balloonmsg** specifies the string to show on the pop-up balloon window; **−statusmsg** specifies the string to show on the status bar; **−msg** specifies a string to show on both the balloon window and the stats bar window. When used together, the **−msg** option has a lower precedence than the **−balloonmsg** and **−statusmsg** options.

The **bind** command can also be used to change the messages after the initial bindings were set. Example:

```
button .b
tixBalloon .bal

# Add balloon binding
.bal bind .b -msg "This is a button"

...

# Change the balloon binding
.bal bind .b -msg "This is a useful button"
```

pathName **cget** *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the **tixBalloon** command.

pathName **configure** ?*option*? *?value option value ...*?

Query or modify the configuration options of the widget. If no *option* is specified, returns a list describing all of the available options for *pathName* (see **Tk_ConfigureInfo** for information on the format of this list). If *option* is specified with no *value*, then the command returns a list describing the one named option (this list will be identical to the corresponding sublist of the value returned if no *option* is specified). If one or more *option−value* pairs are specified, then the command modifies the given widget option(s) to have the given value(s); in this case the command returns an empty string. *Option* may have any of the values accepted by the **tixBalloon** command.

pathName **unbind** *widget*

Cancels the Balloon widget's binding with *widget*.

pathName **subwidget** *name ?args?*

When no options are given, this command returns the pathname of the subwidget of the specified name.

When options are given, the widget command of the specified subwidget will be called with these options.

BINDINGS

After a widget has be bound to a Balloon widget, when the user moves the cursor into this widget, the Balloon widget is activated: if the **−balloonmsg** option of this widget is set, the balloon window pops up; if the **−statusmsg** option of this widget is set, the message will be displayed in the status bar widget.

When the user moves the cursor out of the widget, the Balloon widget is de-activated: the balloon window is withdrawn and the status-bar message removed.

KEYWORDS

Tix(n)

NAME

tixButtonBox − Create and manipulate Tix ButtonBox widgets

SYNOPSIS

tixButtonBox *pathName ?options*?

STANDARD OPTIONS

anchor	**background**	**cursor**
relief	**borderWidth**	

See the **options(n)** manual entry for details on the standard options.

WIDGET-SPECIFIC OPTIONS

Name: **orientation**
Class: **Orientation**
Switch: **−orientation**
Alias: **−orient**

Static Option. Specifies the orientation of the button subwidgets. Only the values "horizontal" and "vertical" are recognized.

Name: **padx**
Class: **Pad**
Switch: **−padx**

Specifies the horizontal padding between two neighboring button subwidgets in the ButtonBox widget.

Name: **pady**
Class: **Pad**
Switch: **−pady**

Specifies the vertical padding between two neighboring button subwidgets in the ButtonBox widget.

Name: **state**
Class: **State**
Switch: **−state**

Specifies the state of all the buttons inside the ButtonBox widget.

Note: Setting this option using the *config* widget command will enable or disable all the buttons subwidgets. Original states of the individual buttons are *not* saved. Only the values "normal" and "disabled" are recognized.

SUBWIDGETS

All the button subwidgets created as a result of the **add** command can be accessed by the **subwidget** command. They are identified by the **buttonName** parameter to the **add** command. Here is an example:

```
tixButtonBox .bbox
pack .bbox
.bbox add eat   -text Eat
.bbox add sleep -text Sleep
.bbox subwidget eat   config -fg green
.bbox subwidget sleep config -fg red
```

DESCRIPTION

The **tixButtonBox** command creates a new window (given by the *pathName* argument) and makes it into a ButtonBox widget. Additional options, described above, may be specified on the command line or in the option database to configure aspects of the ButtonBox such as its cursor and relief.

The ButtonBox widget can be used as a container widget to hold the ‘‘action’’ buttons in a dialog box.

WIDGET COMMAND

The **tixButtonBox** command creates a new Tcl command whose name is the same as the path name of the ButtonBox’s window. This command may be used to invoke various operations on the widget. It has the following general form:

pathName option ?*arg arg ...*?

PathName is the name of the command, which is the same as the ButtonBox widget’s path name. *Option* and the *arg*s determine the exact behavior of the command. The following commands are possible for ButtonBox widgets:

pathName **add** *buttonName* ?*option value ...*?

Add a new button subwidget with the name *buttonName* into the ButtonBox widget. Additional configuration options can be given to configure the new button subwidget.

pathName **cget** *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the **tixButtonBox** command.

pathName **configure** ?*option*? *?value option value ...*?

Query or modify the configuration options of the widget. If no *option* is specified, returns a list describing all of the available options for *pathName* (see **Tk_ConfigureInfo** for information on the format of this list). If *option* is specified with no *value*, then the command returns a list describing the one named option (this list will be identical to the corresponding sublist of the value returned if no *option* is specified). If one or more *option−value* pairs are specified, then the command modifies the given widget option(s) to have the given value(s); in this case the command returns an empty string. *Option* may have any of the values accepted by the **tixButtonBox** command.

pathName **invoke** *buttonName*

Invoke the button subwidget with the name buttonName.

pathName **subwidget** *name ?args?*

When no additional arguments are given, returns the pathname of the subwidget of the specified name.

When no additional arguments are given, the widget command of the specified subwidget will be called with these parameters.

BINDINGS

TixButtonBox widgets have no default bindings. The button subwidgets retain their default Tk bindings.

KEYWORDS

Tix(n), Container Widgets

NAME

tixComboBox - Create and manipulate tixComboBox widgets

SYNOPSIS

tixComboBox *pathName ?options*?

SUPER-CLASS

The **TixComboBox** class is derived from the **TixLabelWidget** class and inherits all the commands, options and subwidgets of its super-class.

STANDARD OPTIONS

TixComboBox supports all the standard options of a frame widget. See the options(n) manual entry for details on the standard options.

WIDGET-SPECIFIC OPTIONS

Name: **anchor**
Class: **Anchor**
Switch: **−anchor**

Specifies how the string inside the entry subwidget should be aligned. Only the values "w" or "e" are allowed. When set the "w", the entry is aligned to its beginning. When set to "e", it is aligned to its end.

Name: **arrowBitmap**
Class: **ArrowBitmap**
Switch: **−arrowbitmap**

Specifies the bitmap to be used in the arrow button beside the entry widget. The default is an downward arrow bitmap in the file $tix_library/bitmaps/cbxarrow

Name: **browseCmd**
Class: **BrowseCmd**
Switch: **−browsecmd**

Specifies the command to be called when the user browses through the listbox. This command can be used to provide instant feedback when the user examines items in the listbox before committing a choice.

Name: **command**
Class: **Command**
Switch: **−command**

Specifies the command to be called when the ComboBox is invoked or when the **−value** of the ComboBox is changed.

Name: **crossBitmap**
Class: **CrossBitmap**
Switch: **−crossbitmap**

Specifies the bitmap to be used in the "cross" button to the left of the entry widget. The default is a bitmap in the file $tix_library/bitmaps/cross

Name: **disableCallback**
Class: **DisableCallback**
Switch: **−disablecallback**

A boolean value indicating whether callbacks should be disabled. When set to true, the TCL command specified by the **−command** option is not executed when the **−value** of the ComboBox. changes.

Name: **disabledforeground**

Class: **DisabledForeground**
Switch: **–disabledforeground**

Specifies the foreground color to be used when the ComboBox is disabled.

Name: **dropdown**
Class: **Dropdown**
Switch: **–dropdown**

A Boolean value specifying the style of the ComboBox. When set to "true", the listbox is only displayed temporarily when the arrow button is pressed. When set to "false", the listbox is always displayed.

Name: **editable**
Class: **Editable**
Switch: **–editable**

Specifies whether the user is allowed to type into the entry subwidget of the ComboBox.

Name: **fancy**
Class: **Fancy**
Switch: **–fancy**

A Boolean value specifying whether the cross and tick button subwidgets should be shown.

Name: **grab**
Class: **Grab**
Switch: **–grab**

Specifies the pointer grabbing policy when the listbox is popped up. Only values "global", "local" or "none" are allowed. By default global grab is used. However, when you are developing your application, you may want to use only local grabbing so that in the event of errors, your X display won´t be locked up.

Name: **historyLimit**
Class: **historyLimit**
Switch: **–historylimit**
Alias: **–histlimit**

Specifies how many previous user inputs can be stored in the history list.

Name: **history**
Class: **History**
Switch: **–history**

A Boolean value specifying whether previous user inputs should be stored in the history list.

Name: **label**
Class: **Label**
Switch: **–label**

Specifies the string to display as the label of this ComboBox widget.

Name: **labelSide**
Class: **LabelSide**
Switch: **–labelside**

Specifies where the label should be displayed relative to the entry subwidget. Valid options are: **top**, **left**, **right**, **bottom**, **none** or **acrosstop**.

Name: **listCmd**
Class: **listCmd**
Switch: **–listcmd**

Specifies a TCL command to be called every time when the listbox pops up. This option allows you to fill up the listbox on-demand. This option is ignored when the listbox is not in the **dropdown** style.

Name: **listWidth**
Class: **listWidth**
Switch: **–listwidth**

If set, this option controls the width of the listbox subwidget when it is popped up. The option is ignored when the listbox is not in the **dropdown** style.

Name: **prunehistory**
Class: **PruneHistory**
Switch: **–prunehistory**

Specifies whether duplicated previous user inputs should be pruned from the the history list. Only Boolean values are allowed.

Name: **selection**
Class: **Selection**
Switch: **–selection**

Contains the selection in the ComboBox (the string displayed in the entry subwidget). Depending on the **–selectmode**, the selection of a ComboBox may be different than its **–value**.

Name: **selection**
Class: **Selection**
Switch: **–selection**

This option stores the temporary selection. When the user types in a text string inside the entry widget, that string is considered as a temporary input and is stored inside the **–selection** option. The **–value** option is updated only when the user presses the return key.

Name: **selectMode**
Class: **SelectMode**
Switch: **–selectmode**

Specifies the how the combobox responds to the mouse button events in the listbox subwidget; can eithet be **"browse"** or **"immediate"**. The default **–selectmode** is "browse". See the **BINDINGS** section below.

Name: **state**
Class: **State**
Switch: **–state**

Specifies the whether the ComboBox is normal or disabled. Only the values "normal" and "disabled" are recognized.

Name: **tickBitmap**
Class: **tickBitmap**
Switch: **–tickbitmap**

Specifies the bitmap to be used in the "tick" button to the left of the entry widget. The default is a bitmap in the file $tix_library/bitmaps/tick

Name: **validateCmd**
Class: **ValidateCmd**
Switch: **–validatecmd**

Specifies a TCL command to be called when the **–value** of the ComboBox is about to change. This command is called with one parameter -- the new **–value** entered by the user. This command is to validate this new value by returning a value it deems valid.

Name: **value**
Class: **Value**
Switch: **−value**

Specifies the string to be displayed in the entry subwidget of the ComboBox. When queried, the returned value is the last value selected by the user. When the **−value** option is changed as a result of the **config -value** widget command, the TCL command specified by the **−command** option is called.

Name: **variable**
Class: **Variable**
Switch: **−variable**

Specifies the global variable in which the value of the ComboBox should be stored. The value of the ComboBox will be automatically updated when this variable is changed.

SUBWIDGETS

Name: **arrow**
Class: **Button**

The down arrow button.

Name: **cross**
Class: **Button**

The cross button. Available only when **−fancy** is set.

Name: **entry**
Class: **Entry**

The entry that shows the value of this **tixControl**.

Name: **label**
Class: **Label**

The label subwidget.

Name: **listbox**
Class: **Listbox**

The listbox that holds all the list entries.

Name: **slistbox**
Class: **TixScrolledListBox**

The scrolled-listbox that provides the scrollbars.

Name: **tick**
Class: **Button**

The tick button. Available only when **−fancy** is set.

DESCRIPTION

The **tixComboBox** command creates a new window (given by the *pathName* argument) and makes it into a **tixComboBox** widget. Additional options, described above, may be specified on the command line or in the option database to configure aspects of the ComboBox such as its cursor and relief.

The Tix ComboBox widget is similar to the combo box control in MS Windows. The user can select a choice by either typing in the entry subwdget or selecting from the listbox subwidget.

WIDGET COMMANDS

The **tixComboBox** command creates a new Tcl command whose name is the same as the path name of the ComboBox's window. This command may be used to invoke various operations on the widget. It has the following general form:

pathName option ?*arg arg ...*?

PathName is the name of the command, which is the same as the ComboBox widget's path name. *Option* and the *arg*s determine the exact behavior of the command. The following commands are possible for ComboBox widgets:

pathName **addhistory** *string*
: Add the string to the beinning of the listbox.

pathName **appendhistory** *string*
: Append the string to the end of the listbox.

pathName **cget** *option*
: Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the **tixComboBox** command.

pathName **configure** ?*option*? *?value option value ...*?
: Query or modify the configuration options of the widget. If no *option* is specified, returns a list describing all of the available options for *pathName* (see **Tk_ConfigureInfo** for information on the format of this list). If *option* is specified with no *value*, then the command returns a list describing the one named option (this list will be identical to the corresponding sublist of the value returned if no *option* is specified). If one or more *option−value* pairs are specified, then the command modifies the given widget option(s) to have the given value(s); in this case the command returns an empty string. *Option* may have any of the values accepted by the **tixComboBox** command.

pathName **flash** *index string*
: Flashes the ComboBox. **flash** is usually called by a −*command* procedure to acknowledge to the user that he has selected a value for the ComboBox.

pathName **insert** *index string*
: Insert the *string* into the listbox at the specified index. *index* must be a valid listbox index.

pathName **pick** *index*
: Set the (*index*)th item in the listbox to be the current value of the ComboBox. As a result, the *value* of the ComboBox is changed and the TCL command sepcified by the −*command* option will be called.

pathName **subwidget** *name ?args?*
: When no options are given, returns the pathname of the subwidget of the specified name.

  When options are given, the widget command of the specified subwidget will be called with these options.

BINDINGS

[1] If the **−selectmode** is "immediate", when the user enters a keystroke, clicks on an item or drags the mouse pointer in the listbox, the **−value** of the ComboBox will be immediately set to this item and the **−command** procedure will be called.

[2] If the **−selectmode** is "browse", when the user enters a keystroke, clicks on an item or drags the mouse pointer in the listbox, the **−selection** of the ComboBox will be immediately set to the new content of the entry subwidget; also the **−browsecmd** procedure will be called. The **−value** option will be changed only when the user invokes the ComboBox by double-clicking on the listbox item, pressing the <Return> or <Tab> key inside the entry subwidget, or switching the input focus to another widget inside the same toplevel widget. If the user presses the <Escape> key at any time, any new **−selection** will be ignored and the text inside the entry subwidget will be restored to the

current **−value** of the ComboBox.

BUGS

Starting from Tix vetsion 4.0, the default **−value** of the ComboBox is the empty string. If you want the ComboBox to show a string by default, you must configure its **−value** option explicitly.

KEYWORDS

Tix(n), ComboBox(n), listbox(n)

NAME

tixControl − Create and manipulate tixControl widgets

SYNOPSIS

tixControl *pathName ?options*?

SUPER-CLASS

The **TixControl** class is derived from the **TixLabelWidget** class and inherits all the commands, options and subwidgets of its super-class.

STANDARD OPTIONS

The Control widget supports all the standard options of a frame widget. See the **options(n)** manual entry for details on the standard options.

WIDGET-SPECIFIC OPTIONS

Name: **allowEmpty**
Class: **AllowEmpty**
Switch: **−allowempty**

Specifies whether the Control widget should allow the empty string as a valid input.

Name: **autorepeat**
Class: **AutoRepeat**
Switch: **−autorepeat**

Specifies whether the Control widget should have autorepeat behavior. If set to be "true", the value of the Control widget will be automatically incremented or decremented when the user holds down the mouse button over the arrow buttons. Only values "true" and "false" will be recognized.

Name: **command**
Class: **Command**
Switch: **−command**

Specifies the command to be called when the **−value** option of the Control widget is changed. The command will be called with one arguments -- the new value of the Control widget.

Name: **decrCmd**
Class: **DecrCmd**
Switch: **−decrcmd**

Specifies a TCL command to be called when the the user presses the down-arrow button subwidget. This command is called with one parameter -- the current **−value** of this Control widget. This command is to decrement this value by one step, according to its own definition of "decrement", and return the decremented value, which will be stored in the **−value** of this Control widget.

Name: **disableCallback**
Class: **DisableCallback**
Switch: **−disablecallback**

A boolean value indicating whether callbacks should be disabled. When set to true, the TCL command specified by the **−command** option is not executed when the **−value** of the Control widget changes.

Name: **disableForeground**
Class: **DisableForeground**
Switch: **−disableforeground**

The foreground color to use for of the entry subwidget when the Control widget is disabled.

Name: **incrCmd**
Class: **IncrCmd**
Switch: **−incrcmd**

Specifies a TCL command to be called when the the user presses the up-arrow button subwidget. This command is called with one parameter -- the current **–value** of this Control widget. This command is to increment this value by one step, according to its own definition of "increment", and return the incremented value, which will be stored in the **–value** of this Control widget.

Name: **initwait**
Class: **Initwait**
Switch: **–initwait**

Specifies how long the Control widget should wait initially before it starts to automatically increment or decrement its value in the autorepeat mode. In milliseconds.

Name: **integer**
Class: **Integer**
Switch: **–integer**

A Boolean value specifying whether only integer numbers are accepted.

Name: **label**
Class: **Label**
Switch: **–label**

Specifies the string to display as the label of this Control widget.

Name: **labelSide**
Class: **LabelSide**
Switch: **–labelside**

Specifies where the label should be displayed relative to the entry subwidget. Valid options are: **top**, **left**, **right**, **bottom**, **none** or **acrosstop**.

Name: **max**
Class: **Max**
Switch: **–max**
Alias: **–ulimit**

Specifies the upper limit of the value of the Control widget. When set to empty string, the Control widget has no upper limit.

Name: **min**
Class: **Min**
Switch: **–min**
Alias: **–llimit**

Specifies the lower limit of the value of the Control widget.When set to empty string, the Control widget has no lower limit.

Name: **repeatRate**
Class: **RepeatRate**
Switch: **–repeatrate**

Specifies how often the value of the Control widget should be incremented or decremented when it is in the autorepeat mode. In milliseconds.

Name: **selectMode**
Class: **SelectMode**
Switch: **–selectmode**

Specifies how the Control widget should react to `<KeyPress>` events. When set to "immediate", any user keyboard inputs will immediately change the **–value** option. When set to "normal", the user keyboard inputs will be copied to the **–value** option only if the `<Return>` key is pressed or the keyboard focus is changed. The use of the immediate mode is discouraged. For effective use of the Control widget, one should use the normal mode together with the **update** widget command (see below).

Name: **state**
Class: **State**
Switch: **–state**

Specifies the whether the Control widget is normal or disabled. Only the values "normal" and "disabled" are recognized.

Name: **step**
Class: **Step**
Switch: **–step**

Specifies by how much the value of the Control widget should be incremented or decrmented when the user press the arrow buttons.

Name: **validateCmd**
Class: **ValidateCmd**
Switch: **–validatecmd**

Specifies a TCL command to be called when the -value of the Control widget is about to change. This command is called with one parameter -- the new **–value** entered by the user. This command is to validate this new value by returning a value it deems valid.

Name: **value**
Class: **Value**
Switch: **–value**

Specifies the value of the Control widget.

Name: **variable**
Class: **Variable**
Switch: **–variable**

Specifies the global variable in which the value of the Control widget should be stored. The value of the Control widget will be automatically updated when this variable is changed.

SUBWIDGETS

Name: **decr**
Class: **Button**

The down arrow button.

Name: **entry**
Class: **Entry**

The entry that shows the value of this Control widget.

Name: **incr**
Class: **Button**

The up arrow button.

Name: **label**
Class: **Label**

The label subwidget.

DESCRIPTION

The **tixControl** command creates a new window (given by the *pathName* argument) and makes it into a Control widget. Additional options, described above, may be specified on the command line or in the option database to configure aspects of the Control widget such as its cursor and relief.

The Control widget is also known as the **SpinBox** widget. It is generally used to control a value. The user can adjust the value by pressing the two arrow buttons or by entering the value directly into the entry. The new value will be checked against the user-defined upper and lower limits.

WIDGET COMMANDS

The **tixControl** command creates a new Tcl command whose name is the same as the path name of the Control widget's window. This command may be used to invoke various operations on the widget. It has the following general form:

pathName option ?*arg arg ...*?

PathName is the name of the command, which is the same as the Control widget's path name. *Option* and the *arg*s determine the exact behavior of the command. The following commands are possible for Control widgets:

pathName **cget** *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the **tixControl** command.

pathName **configure** ?*option*? *?value option value ...*?

Query or modify the configuration options of the widget. If no *option* is specified, returns a list describing all of the available options for *pathName* (see **Tk_ConfigureInfo** for information on the format of this list). If *option* is specified with no *value*, then the command returns a list describing the one named option (this list will be identical to the corresponding sublist of the value returned if no *option* is specified). If one or more *option−value* pairs are specified, then the command modifies the given widget option(s) to have the given value(s); in this case the command returns an empty string. *Option* may have any of the values accepted by the **tixControl** command.

pathName **decr**

Decrements the value of the Control widget by the step specified by the −*step* option.

pathName **incr**

Increments the value of the Control widget by the step specified by the −*step* option.

pathName **invoke**

Causes the command specified by the −*command* option to be invoked.

pathName **update**

If the user has modified the entry using keyboard inputs, the update command will **update** the **−value** of this Control widget. When the Control widget's **−selectmode** option is set to "normal", one should call the **update** command on this widget before examining its **−value** option. This command has no effect in if the **−selectmode** option is set to "immediate".

pathName **subwidget** *name ?args?*

When no options are given, this command returns the pathname of the subwidget of the specified name.

When options are given, the widget command of the specified subwidget will be called with these options.

BINDINGS

When the user presses the up/down arrow buttons (or press the <Up> and <Down> arrow keys on the keyboard), the value of the tixControl widget is adjusted according to the **−validatecmd**, **−incrcmd**, **−decrcmd**, **−step**, **−max** and **−min** options.

KEYWORDS

Tix(n)

NAME

Tix Display Items

DESCRIPTION

The Tix **Display Items** and **Display Types** are devised to solve a general problem: many Tix widgets (both existing and planned ones) display many items of many types simutaneously.

For example, a hierarchical listbox widget (HList) can display items of images, plain text and subwindows in the form of a hierarchy. Another widget, the tabular listbox, (TList, currently planned and will be released in Tix 4.1) also display items of the same types, although it arranges the items in a tabular form. Yet another widget, the spreadsheet widget, also displays similar types items, but in yet another format.

In these examples, the display items in different widgets are only different in how they are arranged by the **host widget**. In Tix, display items are clearly separated from the host widgets. The advantage is two-fold: first, the creation and configuration of display items become uniform across different host widgets. Second, new display item types can be added without the need to modify the existing host widgets.

In a way, Tix display items are similar to the items inside Tk the canvas widget. However, unlike the Tix display items, the canvas items are not independent of the canvas widget; this makes it impossible to use the canvas items inside other types of TK widgets.

The appearance of a display item is controlled by a set of *attributes*. It is observed that each the attributes usually fall into one of two categroies: "*individual*" or "*collective*". For example, the text items inside a HList widget may all display a different text string; however, in most cases, the text items share the same color, font and spacing. Instead of keeping a duplicated version of the same attributes inside each display item, it will be advantageous to put the collective attributes in a special object called a **display style**. First, there is the space concern: a host widget may have many thousands of items; keeping dupilcated attributes will be very wasteful. Second, when it becomes necessary to change a collective attribute, such as changing all the text items' foreground color to red, it will be more efficient to change only the display style object than to modify all the text items one by one.

The attributes of the a display item are thus stored in two places: it has a set of **item options** to store its individual attributes. Each display item is also associated with a *display style*, which specifies the collective attributes of all items associated with itself.

The division between the individual and collective attributes are fixed and cannot be changed. Thus, when it becomes necessary for some items to differ in their collective attributes, two or more **display styles** can be used. For example, suppose you want to display two columns of text items inside an HList widget, one column in red and the other in blue. You can create a TextStyle object called "red", which defines a red foreground, and another called "blue", which defines a blue foreground. You can then associate all text items of the first column to "red" and the second column to "blue".

DISPLAY ITEM TYPES AND OPTIONS

Currently there are three types of display items: **text**, **imagetext** and **window**.

IMAGETEXT ITEMS

Display items of the type **imagetext** are used to display an image together with a text string. Imagetext items support the following options:

ITEM OPTIONS

Name: **bitmap**
Class: **Bitmap**
Switch: **−bitmap**

Specifies the bitmap to display in the item.

Name: **image**
Class: **Image**

Switch: **–image**

Specifies the image to display in the item. When both the **–bitmap** and **–image** options are specified, only the image will be displayed.

Name: **imageTextStyle**
Class: **ImageTextStyle**
Switch: **–style**

Specifies the display style to use for this item. Must be the name of a **imagetext** display style that has already be created by the **tixDisplayStyle(n)** command.

Name: **showImage**
Class: **ShowImage**
Switch: **–showimage**

A Boolean value that specifies whether the image/bitmap should be displayed.

Name: **showText**
Class: **ShowText**
Switch: **–showtext**

A Boolean value that specifies whether the text string should be displayed.

Name: **text**
Class: **Text**
Switch: **–text**

Specifies the text string to display in the item.

Name: **underline**
Class: **Underline**
Switch: **–underline**

Specifies the integer index of a character to underline in the text string in the item. 0 corresponds to the first character of the text displayed in the widget, 1 to the next character, and so on.

STYLE OPTIONS

The style information of **imagetext** items are stored in the **imagetext** display style. The following options are supported:

STANDARD OPTIONS

```
activeBackground        activeForeground
anchor                  background
disabledBackground      disabledForeground
foreground              font
justify                 padX
padY                    selectBackground
selectForeground        wrapLength
```

See the **options(n)** manual entry for details on the standard options.

STYLE-SPECIFIC OPTIONS

Name: **gap**
Class: **Gap**
Switch: **–gap**

Specifies the distance between the bitmap/image and the text string, in number of pixels.

TEXT ITEMS

Display items of the type **text** are used to display a text string in a widget. Text items support the following options:

ITEM OPTIONS

Name: **textStyle**
Class: **TextStyle**
Switch: **–style**

Specifies the display style to use for this text item. Must be the name of a **text** display style that has already be created by the **tixDisplayStyle(n)** command.

Name: **text**
Class: **Text**
Switch: **–text**

Specifies the text string to display in the item.

Name: **underline**
Class: **Underline**
Switch: **–underline**

Specifies the integer index of a character to underline in the item. 0 corresponds to the first character of the text displayed in the widget, 1 to the next character, and so on.

STYLE OPTIONS

STANDARD OPTIONS

```
activeBackground          activeForeground
anchor                    background
disabledBackground        disabledForeground
foreground                font
justify                   padX
padY                      selectBackground
selectForeground          wrapLength
```

See the **options(n)** manual entry for details on the standard options.

WINDOW ITEMS

Display items of the type **window** are used to display a sub-window in a widget. **Window** items support the following options:

ITEM OPTIONS

Name: **windowStyle**
Class: **WindowStyle**
Switch: **–style**

Specifies the display style to use for this window item. Must be the name of a **window** display style that has already be created by the **tixDisplayStyle(n)** command.

Name: **window**
Class: **Window**
Switch: **–window**
Alias: **–widget**

Specifies the sub-window to display in the item.

STYLE OPTIONS

STANDARD OPTIONS

```
anchor
padX                             padY
```

See the **options(n)** manual entry for details on the standard options.

CREATING DISPLAY ITEMS

Display items do not exist on their and thus they cannot be created independently of the widgets they reside in. As a rule, display items are created by special widget commands of their "host" widgets. For example, the HList widgets has a command **item** which can be used to create new display items. The following code creates a new imagetext item at the third column of the entry foo inside an HList widget:

```
tixHList .h −columns 3
.h add foo
.h item create foo 2 −itemtype imagetext −text Hello −image image1
```

The **item create** command of the HList widget accepts a variable number of arguments. The special argument **−itemtype** specifies which type of display item to create. Options that are valid for this type of display items can then be specified by one or more *option−value* pairs.

After the display item is created, they can then be configured or destroyed using the commands provided by the host widget. For example, the HList widget has the command **item configure**, **item cget** and **item delete** for accessing the display items.

CREATING AND MANIPULATING DISPLAY STYLES

Display styles are created by the command **tixDisplayStyle**:

SYNOPSIS

tixDisplayStyle *itemType* ?−*stylename name*? ?−*refwindow pathName*? ?*options value ...*?

itemType must be one of the existing display items types such as **text**, **imagetext**, **window** or any new types added by the user. Additional arguments can be given in one or more *option−value* pairs. *option* can be any of the valid option for this display style or any of the following:

−stylename *name*
: Specifies a name for this style. If unspecified, then a default name will be chosen for this style.

−refwindow *pathName*
: Specifies a window to use for determine the default values of the display type. If unspecified, the main window will be used. Default values for the display types can be set via the options database. The following example sets the **−disablebackground** and **−disabledforeground** options of a **text** display style via the option database:

```
option add *table.list*disabledForeground blue
option add *table.list*disabledBackground darkgray
tixDisplayStyle text −refwindow .table.list −fg red
```

By using the option database to set the options of the display styles, we can advoid hardcoding the option values and give the user more flexibility in customization. See option(n) for a detailed description of the option database.

STYLE COMMAND

The **tixDisplayStyle** command creates a new Tcl command whose name is the same as the name of the newly created display style. This command may be used to invoke various operations on the display style. It has the following general form:

styleName option ?*arg arg ...*?

styleName is the name of the command. *Option* and the *arg*s determine the exact behavior of the command. The following commands are possible:

styleName **cget** *option*
: Returns the current value of the configuration option given by *option*. *Option* may have any of the valid options of this display style.

styleName **configure** ?*option*? ?*value option value ...*?
: Query or modify the configuration options of the display style. If no *option* is specified, returns a list describing all of the available options for *styleName* (see **Tk_ConfigureInfo** for information on the format of this list). If *option* is specified with no *value*, then the command returns a list describing the one named option (this list will be identical to the corresponding sublist of the value returned if no *option* is specified). If one or more *option−value* pairs are specified, then the command modifies the given option(s) to have the given value(s); in this case the command returns an empty string. *Option* may have any of the valid options of this display style.

styleName **delete**
: Destroy this display style object.

EXAMPLE

The following example creates two columns of data in a HList widget. The first column is in red and the second column in blue. The colors of the columns are controlled by two different **text** styles. Also, the anchor and font of the second column is chosen so that the income data is aligned properly.

```
set courier -*-courier-medium-r-*-*-14-*-*-*-*-*-*-*
tixHList .h -columns 2; pack .h
set red  [tixDisplayStyle text -fg #800000]
set blue [tixDisplayStyle text -fg #000080 -anchor e -font $courier]

foreach n {{Joe $10,000} {Peter $20,000} {Raj $90,000} {Zinh $0}} {
    set entry [.h addchild {}]
    .h item create $entry 0 -itemtype text \
                   -text [lindex $n 0] -style $red
    .h item create $entry 1 -itemtype text \
                   -text [lindex $n 1] -style $blue
}
```

NAME

tixDestroy − Destroy Tix Objects

SYNOPSIS

tixDestroy *objectName*

DESCRIPTION

The **tixDestroy** destroys a Tix object whose class is declared by the **tixClass** keyword. When the object is destroyed, its **Destructor** function is called and the memory allocated for this object is freed.

KEYWORDS

Tix, Object

NAME

tixDirList – Create and manipulate tixDirList widgets

SYNOPSIS

tixDirList *pathName ?options*?

SUPER-CLASS

The **TixDirList** class is derived from the **TixScrolledHList** class and inherits all the commands, options and subwidgets of its super-class.

STANDARD OPTIONS

TixDirList supports all the standard options of a frame widget. See the **options(n)** manual entry for details on the standard options.

WIDGET-SPECIFIC OPTIONS

Name: **browseCmd**
Class: **BrowseCmd**
Switch: **–browsecmd**

Specifies a command to call whenever the user browses on a directory (usually by single-clicking on the name of the directory). The command is called with one argument, the complete pathname of the directory.

Name: **command**
Class: **Command**
Switch: **–command**

Specifies the command to be called when the user activates on a directory (usually by double-clicking on the name of the directory). The command is called with one argument, the complete pathname of the directory.

Name: **dircmd**
Class: **DirCmd**
Switch: **–dircmd**

Specifies the TCL command to be called when a directory listing is needed for a particular directory. If this option is not specified, by default the DirList widget will attempt to read the directory as a Unix directory. On special occasions, the application programmer may want to supply a special method for reading directories: for example, when he needs to list remote directories. In this case, the **–dircmd** option can be used. The specified command accepts two arguments: the first is the name of the directory to be listed; the second is a Boolean value indicating whether hidden sub-directories should be listed. This command returns a list of names of the sub-directories of this directory. For example:

```
proc read_dir {dir show_hidden} {
    if {$dir == "C:\"} {
        return {DOS NORTON WINDOWS}
    } else {
        return {}
    }
}
```

Name: **disableCallback**
Class: **DisableCallback**
Switch: **–disablecallback**

A boolean value indicating whether callbacks should be disabled. When set to true, the TCL command specified by the **−command** option is not executed when the **−value** of the DirList widget changes.

Name: **showHidden**
Class: **ShowHidden**
Switch: **−showhidden**

Specifies whether hidden directories should be shown. By default, a directory name starting with a period "." is considered as a hidden directory. This rule can be overridden by supplying an alternative **−dircmd** option.

Name: **root**
Class: **Root**
Switch: **−root**

Specifies the name of the root directory. Usually this is **"/"** under Unix machines, but can be changed to **"C:\"** in DOS environments.

Name: **rootName**
Class: **RootNAme**
Switch: **−rootname**

Specifies a text string to display at the root directory. If unspecified, the text string will be the same as the string specified by **−root**.

Name: **value**
Class: **Value**
Switch: **−value**
Alias: **−directory**

Specifies the name of the current directory to be displayed in the DirList widget.

SUBWIDGETS

Name: **hlist**
Class: **TixHList**

The hierarchical listbox that displays the directory listing.

Name: **hsb**
Class: **Scrollbar**

The horizontal scrollbar subwidget.

Name: **vsb**
Class: **Scrollbar**

The vertical scrollbar subwidget.

DESCRIPTION

The **tixDirList** command creates a new window (given by the *pathName* argument) and makes it into a DirList widget. Additional options, described above, may be specified on the command line or in the option database to configure aspects of the DirList such as its cursor and relief.

The DirList widget displays a list view of a directory, its previous directories and its sub-directories. The user can choose one of the directories displayed in the list or change to another directory.

WIDGET COMMANDS

The **tixDirList** command creates a new Tcl command whose name is the same as the path name of the DirList's window. This command may be used to invoke various operations on the widget. It has the following general form:

pathName option ?*arg arg ...*?

PathName is the name of the command, which is the same as the DirList widget's path name. *Option* and the *arg*s determine the exact behavior of the command. The following commands are possible for DirList widgets:

pathName **cget** *option*
Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the **tixDirList** command.

pathName **chdir** *dir*
Change the current directory to *dir*.

pathName **configure** ?*option*? *?value option value ...*?
Query or modify the configuration options of the widget. If no *option* is specified, returns a list describing all of the available options for *pathName* (see **Tk_ConfigureInfo** for information on the format of this list). If *option* is specified with no *value*, then the command returns a list describing the one named option (this list will be identical to the corresponding sublist of the value returned if no *option* is specified). If one or more *option−value* pairs are specified, then the command modifies the given widget option(s) to have the given value(s); in this case the command returns an empty string. *Option* may have any of the values accepted by the **tixDirList** command.

pathName **subwidget** *name ?args?*
When no options are given, this command returns the pathname of the subwidget of the specified name.

When options are given, the widget command of the specified subwidget will be called with these options.

BINDINGS

The mouse and keyboard bindings of the DirList widget is the same as the bindings of the HList widget.

KEYWORDS

Tix(n)

NAME

tixDirTree − Create and manipulate tixDirTree widgets

SYNOPSIS

tixDirTree *pathName ?options*?

SUPER-CLASS

The **TixDirTree** class is derived from the **TixScrolledHList** class and inherits all the commands, options and subwidgets of its super-class.

STANDARD OPTIONS

TixDirTree supports all the standard options of a frame widget. See the **options(n)** manual entry for details on the standard options.

WIDGET-SPECIFIC OPTIONS

Name: **browseCmd**
Class: **BrowseCmd**
Switch: **−browsecmd**

Specifies a command to call whenever the user browses on a directory (usually by single-clicking on the name of the directory). The command is called with one argument, the complete pathname of the directory.

Name: **command**
Class: **Command**
Switch: **−command**

Specifies the command to be called when the user activates on a directory (usually by double-clicking on the name of the directory). The command is called with one argument, the complete pathname of the directory.

Name: **dircmd**
Class: **DirCmd**
Switch: **−dircmd**

Specifies the TCL command to be called when a directory listing is needed for a particular directory. If this option is not specified, by default the DirTree widget will attempt to read the directory as a Unix directory. On special occasions, the application programmer may want to supply a special method for reading directories: for example, when he needs to list remote directories. In this case, the **−dircmd** option can be used. The specified command accepts two arguments: the first is the name of the directory to be listed; the second is a Boolean value indicating whether hidden sub-directories should be listed. This command returns a list of names of the sub-directories of this directory. For example:

```
proc read_dir {dir show_hidden} {
    if {$dir == "C:\"} {
        return {DOS NORTON WINDOWS}
    } else {
        return {}
    }
}
```

Name: **disableCallback**
Class: **DisableCallback**
Switch: **−disablecallback**

A boolean value indicating whether callbacks should be disabled. When set to true, the TCL command specified by the **−command** option is not executed when the **−value** of the DirTree widget changes.

Name: **showHidden**
Class: **ShowHidden**
Switch: **−showhidden**

Specifies whether hidden directories should be shown. By default, a directory name starting with a period "." is considered as a hidden directory. This rule can be overridden by supplying an alternative **−dircmd** option.

Name: **value**
Class: **Value**
Switch: **−value**
Alias: **−directory**

Specifies the name of the current directory to be displayed in the DirTree widget.

SUBWIDGETS

Name: **hlist**
Class: **TixHList**

The hierarchical listbox that displays the directory listing.

Name: **hsb**
Class: **Scrollbar**

The horizontal scrollbar subwidget.

Name: **vsb**
Class: **Scrollbar**

The vertical scrollbar subwidget.

DESCRIPTION

The **tixDirTree** command creates a new window (given by the *pathName* argument) and makes it into a DirTree widget. Additional options, described above, may be specified on the command line or in the option database to configure aspects of the DirTree such as its cursor and relief.

The DirTree widget displays a list view of a directory, its previous directories and its sub-directories. The user can choose one of the directories displayed in the list or change to another directory.

WIDGET COMMANDS

The **tixDirTree** command creates a new Tcl command whose name is the same as the path name of the DirTree's window. This command may be used to invoke various operations on the widget. It has the following general form:

pathName option ?*arg arg ...*?

PathName is the name of the command, which is the same as the DirTree widget's path name. *Option* and the *arg*s determine the exact behavior of the command. The following commands are possible for DirTree widgets:

pathName **cget** *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the **tixDirTree** command.

pathName **chdir** *dir*

Change the current directory to *dir*.

pathName **configure** ?*option*? ?*value option value* ...?

Query or modify the configuration options of the widget. If no *option* is specified, returns a list describing all of the available options for *pathName* (see **Tk_ConfigureInfo** for information on the format of this list). If *option* is specified with no *value*, then the command returns a list describing the one named option (this list will be identical to the corresponding sublist of the value returned if no *option* is specified). If one or more *option−value* pairs are specified, then the command modifies the given widget option(s) to have the given value(s); in this case the command returns an empty string. *Option* may have any of the values accepted by the **tixDirTree** command.

pathName **subwidget** *name ?args?*

When no options are given, this command returns the pathname of the subwidget of the specified name.

When options are given, the widget command of the specified subwidget will be called with these options.

BINDINGS

The mouse and keyboard bindings of the DirTree widget is the same as the bindings of the HList widget.

KEYWORDS

Tix(n)

NAME

tixExFileSelectBox − Create and manipulate tixExFileSelectBox widgets

SYNOPSIS

tixExFileSelectBox *pathName ?options*?

SUPER-CLASS

The **TixExFileSelectBox** class does not have a super-class.

STANDARD OPTIONS

TixExFileSelectBox supports all the standard options of a frame widget. See the **options(n)** manual entry for details on the standard options.

WIDGET-SPECIFIC OPTIONS

Name: **browseCmd**
Class: **BrowseCmd**
Switch: **−browsecmd**

Specifies a command to call whenever the user browses on a filename in the file listbox (usually by single-clicking on the filename). The command is called with one argument, the complete pathname of the file.

Name: **command**
Class: **Command**
Switch: **−command**

Specifies the command to be called when the user chooses on a filename the file listbox (usually by double-clicking on the filename). The command is called with one argument, the complete pathname of the file.

Name: **dialog**
Class: **Dialog**
Switch: **−dialog**

Specifies a dialog box which contains this ExFileSelectBox widget. The dialog box must be a widget of the class TixShell or its descendant classes. *This is an internal option and should not be used by application programmers.*

Name: **dircmd**
Class: **DirCmd**
Switch: **−dircmd**

Specifies the TCL command to be called when a file listing is needed for a particular directory. If this option is not specified, by default the ExFileSelectBox widget will attempt to read the directory as a Unix directory. On special occasions, the application programmer may want to supply a special method for reading directories: for example, when he needs to list remote files. In this case, the **−dircmd** option can be used. The specified command accepts three arguments: the first is the name of the directory to be listed; the second is a list of file patterns, the third is a Boolean value indicating whether hidden files should be listed. This command returns a list of names of the files of this directory which match with the file patterns.

Name: **directory**
Class: **Directory**
Switch: **−directory**
Alias: **−dir**

Specifies the current directory whose files and sub-directories are displayed in the ExFileSelectBox.

Name: **disableCallback**
Class: **DisableCallback**

Switch: **−disablecallback**

A boolean value indicating whether callbacks should be disabled. When set to true, the TCL command specified by the **−command** option is not executed when the **−value** of the ExFileSelectBox widget changes.

Name: **fileTypes**
Class: **FileTypes**
Switch: **−filetypes**

Specifies the file types that can be selected from the "List Files of Type:" ComboBox subwidget. The value of this option must be a TCL list; each item of this list must in turn be a list of two elements. The first element is a list of file patterns. The second element is a string that describe these file patterns. For example:

```
tixExFileSelectBox .box -filetypes {
    {{*}          {All files}}
    {{*.txt}      {Text files}}
    {{*.c *.h}    {C source files}}
}
```

Name: **showHidden**
Class: **ShowHidden**
Switch: **−showhidden**

Specifies whether hidden directories should be shown. By default, a directory name starting with a period "." is considered as a hidden directory.

Name: **pattern**
Class: **Pattern**
Switch: **−pattern**

Specifies whether the file pattern(s) to match with the files in the current directory. One or more file patterns can be given at the same time. For example, {*.c *.h} will match all files that have either the ".h" or ".c" extensions.

Name: **value**
Class: **Value**
Switch: **−value**
Alias: **−selection**

Specifies the name of the filename currently selected by the user.

SUBWIDGETS

Name: **cancel**
Class: **Button**

The button widget with the "Cancel" label.

Name: **dir**
Class: **TixComboBox**

The ComboBox subwidget under the "Directories" heading.

Name: **dirlist**
Class: **TixDirList**

The DirList subwidget that shows the hierarchical list of directories.

Name: **file**
Class: **TixComboBox**

The ComboBox subwidget under the "Files" heading.

Name: **filelist**
Class: **TixScrolledListBox**

The ScrolledListBox subwidget that shows the list of filenames.

Name: **hidden**
Class: **Checkbutton**

The checkbutton widget with the "Show Hidden Files" label.

Name: **ok**
Class: **Button**

The button widget with the "OK" label.

Name: **types**
Class: **TixComboBox**

The ComboBox subwidget under the "List Files of Type" heading.

DESCRIPTION

The **tixExFileSelectBox** command creates a new window (given by the *pathName* argument) and makes it into a ExFileSelectBox widget. Additional options, described above, may be specified on the command line or in the option database to configure aspects of the ExFileSelectBox such as its cursor and relief.

The ExFileSelectBox widget is usually embedded in a tixExFileSelectDialog widget. It provides an convenient method for the user to select files. The style of the ExFileSelectBox widget is very similar to the standard file dialog in MS Windows 3.1.

WIDGET COMMANDS

The **tixExFileSelectBox** command creates a new Tcl command whose name is the same as the path name of the ExFileSelectBox's window. This command may be used to invoke various operations on the widget. It has the following general form:

pathName option ?*arg arg ...*?

PathName is the name of the command, which is the same as the ExFileSelectBox widget's path name. *Option* and the *arg*s determine the exact behavior of the command. The following commands are possible for ExFileSelectBox widgets:

pathName **cget** *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the **tixExFileSelectBox** command.

pathName **configure** ?*option*? ?*value option value ...*?

Query or modify the configuration options of the widget. If no *option* is specified, returns a list describing all of the available options for *pathName* (see **Tk_ConfigureInfo** for information on the format of this list). If *option* is specified with no *value*, then the command returns a list describing the one named option (this list will be identical to the corresponding sublist of the value returned if no *option* is specified). If one or more *option−value* pairs are specified, then the command modifies the given widget option(s) to have the given value(s); in this case the command returns an empty string. *Option* may have any of the values accepted by the **tixExFileSelectBox** command.

pathName **filter**

Forces the ExFileSelectBox widget to re-filter all the filenames according to the **−pattern** option.

pathName **invoke**

Forces the ExFileSelectBox widget to perform actions as if the user has pressed the "OK" button.

pathName **subwidget** *name ?args?*

When no options are given, this command returns the pathname of the subwidget of the specified name.

When options are given, the widget command of the specified subwidget will be called with these options.

KEYWORDS

Tix(n)

NAME

tixExFileSelectDialog − Create and manipulate tixExFileSelectDialog widgets

SYNOPSIS

tixExFileSelectDialog *pathName ?options*?

SUPER-CLASS

The **TixExFileSelectDialog** class does not have a super-class.

STANDARD OPTIONS

TixExFileSelectDialog supports all the standard options of a frame widget. See the **options(n)** manual entry for details on the standard options.

WIDGET-SPECIFIC OPTIONS

Name: **command**
Class: **Command**
Switch: **−command**

Specifies the command to be called when the user chooses on a filename (usually by selecting the filename and clicking on the "OK" button"). The command is called with one argument, the complete pathname of the file.

SUBWIDGETS

Name: **fsbox**
Class: **TixExFileSelectBox**

The ExFileSelectBox subwidget embedded inside the ExFileSelectDialog.

DESCRIPTION

The **tixExFileSelectDialog** command creates a new window (given by the *pathName* argument) and makes it into a ExFileSelectDialog widget. Additional options, described above, may be specified on the command line or in the option database to configure aspects of the ExFileSelectDialog such as its cursor and relief.

The ExFileSelectDialog widget provides an convenient method for the user to select files. The style of the ExFileSelectDialog widget is very similar to the standard file dialog in MS Windows 3.1.

WIDGET COMMANDS

The **tixExFileSelectDialog** command creates a new Tcl command whose name is the same as the path name of the ExFileSelectDialog's window. This command may be used to invoke various operations on the widget. It has the following general form:

pathName option ?*arg arg ...*?

PathName is the name of the command, which is the same as the ExFileSelectDialog widget's path name. *Option* and the *arg*s determine the exact behavior of the command. The following commands are possible for ExFileSelectDialog widgets:

pathName **cget** *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the **tixExFileSelectDialog** command.

pathName **configure** ?*option*? *?value option value ...*?

Query or modify the configuration options of the widget. If no *option* is specified, returns a list describing all of the available options for *pathName* (see **Tk_ConfigureInfo** for information on the format of this list). If *option* is specified with no *value*, then the command returns a list describing the one named option (this list will be identical to the corresponding sublist of the value returned if no *option* is specified). If one or more *option−value* pairs are specified, then the command modifies the given widget option(s) to have the given value(s); in this case the command returns an empty string. *Option* may have any of the values accepted by the **tixExFileSelectDialog** command.

pathName **popdown**
: Withdraws the ExFileSelectDialog from the screen.

pathName **popup**
: Pops up the ExFileSelectDialog on the screen.

pathName **subwidget** *name ?args?*
: When no options are given, this command returns the pathname of the subwidget of the specified name.

  When options are given, the widget command of the specified subwidget will be called with these options.

KEYWORDS

Tix(n)

NAME

tixFileSelectBox − Create and manipulate Tix FileSelectBox widgets

SYNOPSIS

tixFileSelectBox *pathName ?options*?

STANDARD OPTIONS

The FileSelectBox widget supports all the standard options of a frame widget. See the **options(n)** manual entry for details on the standard options.

WIDGET-SPECIFIC OPTIONS

Name: **browsecmd**
Class: **browseCmd**
Switch: **−browsecmd**

Specifies the command to execute when the user browses through the files. By default, if the **−browsecmd** is specified, the browse command will be executed when the user clicks on a filename in the *Files* listbox.

Name: **command**
Class: **Command**
Switch: **−command**

Specifies the command to execute when the FileSelectBox is invoked. This command is executed with one parameter : the filename selected by the user.

Name: **directory**
Class: **Directory**
Switch: **−directory**
Alias: **−dir**

Specifies the directory to look for files. By default this will be the current working directory of the program and will be changed as the user browses through the directories.

Name: **disableCallback**
Class: **DisableCallback**
Switch: **−disablecallback**

A boolean value indicating whether callbacks should be disabled. When set to true, the TCL command specified by the **−command** option is not executed when the **−value** of the ExFileSelectBox widget changes.

Name: **pattern**
Class: **Pattern**
Switch: **−pattern**

Specifies the matching pattern of the file names that should be listed in the *Files* listbox. For example "*.c" matches all the filenames that end with ".c". If this option is set to the empty string, the default pattern "*" will be used.

Name: **value**
Class: **Value**
Switch: **−value**
Alias: **−selection**

Specifies the name of the filename currently selected by the user.

SUBWIDGETS

Name: **dirlist**
Class: **TixScrolledListBox**

The scrolled listbox that shows the directories.

Name: **filelist**
Class: **TixScrolledListBox**

The scrolled listbox that shows the files.

Name: **filter**
Class: **TixComboBox**

The ComboBox listbox that shows the filter string.

Name: **selection**
Class: **TixComboBox**

The ComboBox listbox that shows the file selection.

DESCRIPTION

The **tixFileSelectBox** command creates a new window (given by the *pathName* argument) and makes it into a FileSelectBox widget. Additional options, described above, may be specified on the command line or in the option database to configure aspects of the FileSelectBox such as its cursor and relief.

The FileSelectBox is similar to the standard Motif(TM) file-selection box. It is generally used for the user to choose a file. FileSelectBox stores the files mostly recently selected into a ComboBox widget so that they can be quickly selected again. The **tixFileSelectDialog** widget is a combination of the FileSelectBox widget and a dialog widget.

WIDGET COMMAND

The **tixFileSelectBox** command creates a new Tcl command whose name is the same as the path name of the FileSelectBox's window. This command may be used to invoke various operations on the widget. It has the following general form:

pathName option ?*arg arg ...*?

PathName is the name of the command, which is the same as the FileSelectBox widget's path name. *Option* and the *arg*s determine the exact behavior of the command. The following commands are possible for FileSelectBox widgets:

pathName **cget** *option*
Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the **tixFileSelectBox** command.

pathName **configure** ?*option*? ?*value option value ...*?
Query or modify the configuration options of the widget. If no *option* is specified, returns a list describing all of the available options for *pathName* (see **Tk_ConfigureInfo** for information on the format of this list). If *option* is specified with no *value*, then the command returns a list describing the one named option (this list will be identical to the corresponding sublist of the value returned if no *option* is specified). If one or more *option−value* pairs are specified, then the command modifies the given widget option(s) to have the given value(s); in this case the command returns an empty string. *Option* may have any of the values accepted by the **tixFileSelectBox** command.

pathName **filter**
Updates the files listed in the FileSelectBox according to the filtering pattern sepcified in the **filter** subwidget.

pathName **invoke**
Execute the command specified by the **−command** option with the filename stored in the **selection** subwidget.

pathName **subwidget** *name ?args?*

When no options are given, this command returns the pathname of the subwidget of the specified name.

When options are given, the widget command of the specified subwidget will be called with these options.

DEFAULT BINDINGS

TIX automatically creates class bindings for FileSelectBoxes that give them the following default behavior:

[1] Mouse button 1 in the *Directory* listbox will change the filter string to the selected directory.

[2] Mouse button 1 in the *Files* listbox will change the filename that appears in the *Selection* entry. It will also trigger the **−browsecmd** if the option has been specified.

[3] The current directory will be changed by (1) double clicking the *Directory* listbox or (2) invoking the *Filter* ComboBox. Please refer to the man page of **tixComboBox** for the default bindings of the ComboBoxes and how they can be invoked.

[4] The command specified by the option -command will be invoked by (1) double clicking the *Files* listbox or (2) invoking *Selection* ComboBox.

KEYWORDS

tixFileSelectBox, tixComboBox, tixFileSelectDialog, Tix(n),

NAME

tixFileSelectDialog − Create and manipulate tixFileSelectDialog widgets

SYNOPSIS

tixFileSelectDialog *pathName ?options*?

SUPER-CLASS

The **TixFileSelectDialog** class does not have a super-class.

STANDARD OPTIONS

TixFileSelectDialog supports all the standard options of a frame widget. See the **options(n)** manual entry for details on the standard options.

WIDGET-SPECIFIC OPTIONS

Name: **command**
Class: **Command**
Switch: **−command**

Specifies the command to be called when the user chooses on a filename (usually by selecting the filename and clicking on the "OK" button"). The command is called with one argument, the complete pathname of the file.

SUBWIDGETS

Name: **btns**
Class: **TixStdButtonBox**

The StdButtonBox subwidget at the bottom of FileSelectDialog. It contains the "OK", "Filter, "Cancel" and "Help" buttons.

Name: **fsbox**
Class: **TixFileSelectBox**

The FileSelectBox subwidget at the top of the FileSelectDialog.

DESCRIPTION

The **tixFileSelectDialog** command creates a new window (given by the *pathName* argument) and makes it into a FileSelectDialog widget. Additional options, described above, may be specified on the command line or in the option database to configure aspects of the FileSelectDialog such as its cursor and relief.

The FileSelectDialog widget provides an convenient method for the user to select files. The FileSelectBox is similar to the standard Motif(TM) file-selection box.

WIDGET COMMANDS

The **tixFileSelectDialog** command creates a new Tcl command whose name is the same as the path name of the FileSelectDialog's window. This command may be used to invoke various operations on the widget. It has the following general form:

pathName option ?*arg arg ...*?

PathName is the name of the command, which is the same as the FileSelectDialog widget's path name. *Option* and the *arg*s determine the exact behavior of the command. The following commands are possible for FileSelectDialog widgets:

pathName **cget** *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the **tixFileSelectDialog** command.

pathName **configure** ?*option*? *?value option value ...*?

Query or modify the configuration options of the widget. If no *option* is specified, returns a list describing all of the available options for *pathName* (see **Tk_ConfigureInfo** for information on the format of this list). If *option* is specified with no *value*, then the command returns a list describing the one named option (this list will be identical to the corresponding sublist of the value returned if no *option* is specified). If one or more *option−value* pairs are specified, then the command modifies the given widget option(s) to have the given value(s); in this case the command returns an empty string. *Option* may have any of the values accepted by the **tixFileSelectDialog** command.

pathName **popdown**

Withdraws the FileSelectDialog from the screen.

pathName **popup**

Pops up the FileSelectDialog on the screen.

pathName **subwidget** *name ?args?*

When no options are given, this command returns the pathname of the subwidget of the specified name.

When options are given, the widget command of the specified subwidget will be called with these options.

KEYWORDS

Tix(n)

NAME

tixFileEntry − Create and manipulate tixFileEntry widgets

SYNOPSIS

tixFileEntry *pathName ?options*?

SUPER-CLASS

The **TixFileEntry** class is derived from the **TixLabelWidget** class and inherits all the commands, options and subwidgets of its super-class.

STANDARD OPTIONS

The FileEntry widget supports all the standard options of a frame widget. See the **options(n)** manual entry for details on the standard options.

WIDGET-SPECIFIC OPTIONS

Name: **activateCmd**
Class: **ActivateCmd**
Switch: **−activatecmd**

Specifies the command to be called when the user activates the **button** subwidget. This command is called before the file dialog is popped up and can be used to customize the file dialog (which may be shared by several FileEnt widget).

Name: **command**
Class: **Command**
Switch: **−command**

Specifies the command to be called when the **−value** option of the FileEntry is changed. This usually happens when the user inputs a filename into the entry subwidget and hits the <Return> key. The command will be called with one arguments -- the new value of the FileEntry widget.

Name: **dialogType**
Class: **DialogType**
Switch: **−dialogtype**

Specifies which type of file selection dialog should be popped up when the user invokes the **button** subwidget. Current only two values are valid: **tixFileSelectDialog** or **tixExFileSelectDialog**.

Name: **disableCallback**
Class: **DisableCallback**
Switch: **−disablecallback**

A boolean value indicating whether callbacks should be disabled. When set to true, the TCL command specified by the **−command** option is not executed when the **−value** of the FileEntry widget changes.

Name: **disableForeground**
Class: **DisableForeground**
Switch: **−disableforeground**

The foreground color to use for of the entry subwidget when the FileEntry widget is disabled.

Name: **fileBitmap**
Class: **FileBitmap**
Switch: **−filebitmap**

Specifies the bitmap to display in side the **button** subwidget.

Name: **label**
Class: **Label**
Switch: **−label**

Specifies the string to display as the label of this FileEntry widget.

Name: **labelSide**
Class: **LabelSide**
Switch: **–labelside**

Specifies where the label should be displayed relative to the entry subwidget. Valid options are: **top**, **left**, **right**, **bottom**, **none** or **acrosstop**.

Name: **selectMode**
Class: **SelectMode**
Switch: **–selectmode**

Specifies how the FileEntry widget should react to `<KeyPress>` events. When set to "immediate", any user keyboard inputs will immediately change the **–value** option. When set to "normal", the user keyboard inputs will be copied to the **–value** option only if the `<Return>` key is pressed or the keyboard focus is changed. The use of the immediate mode is discouraged. For effective use of the FileEntry widget, one should use the normal mode together with the **update** widget command (see below).

Name: **state**
Class: **State**
Switch: **–state**

Specifies the whether the FileEntry widget is normal or disabled. Only the values "normal" and "disabled" are recognized.

Name: **validateCmd**
Class: **ValidateCmd**
Switch: **–validatecmd**

Specifies a TCL command to be called when the -value of the FileEntry widget is about to change. This command is called with one parameter -- the new **–value** entered by the user. This command is to validate this new value by returning a value it deems valid.

Name: **value**
Class: **Value**
Switch: **–value**

Specifies the value of the FileEntry.

Name: **variable**
Class: **Variable**
Switch: **–variable**

Specifies the global variable in which the value of the FileEntry should be stored. The value of the FileEntry will be automatically updated when this variable is changed.

SUBWIDGETS

Name: **button**
Class: **Button**

The button subwidget next to the entry subwidget.

Name: **entry**
Class: **Entry**

The entry subwidget in which the user can type in a filename.

DESCRIPTION

The **tixFileEntry** command creates a new window (given by the *pathName* argument) and makes it into a FileEntry widget. Additional options, described above, may be specified on the command line or in the option database to configure aspects of the FileEntry such as its cursor and relief.

The FileEntry widget can be used to input a filename. The user can type in the filename manually. Alternatively, the user can press the button widget that sits next to the entry, which will bring up a file selection dialog of the type specified by the **−dialogtype** option.

WIDGET COMMANDS

The **tixFileEntry** command creates a new Tcl command whose name is the same as the path name of the FileEntry's window. This command may be used to invoke various operations on the widget. It has the following general form:

> *pathName option* ?*arg arg ...*?

PathName is the name of the command, which is the same as the FileEntry widget's path name. *Option* and the *arg*s determine the exact behavior of the command. The following commands are possible for FileEntry widgets:

pathName **cget** *option*
: Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the **tixFileEntry** command.

pathName **configure** ?*option*? *?value option value ...*?
: Query or modify the configuration options of the widget. If no *option* is specified, returns a list describing all of the available options for *pathName* (see **Tk_ConfigureInfo** for information on the format of this list). If *option* is specified with no *value*, then the command returns a list describing the one named option (this list will be identical to the corresponding sublist of the value returned if no *option* is specified). If one or more *option−value* pairs are specified, then the command modifies the given widget option(s) to have the given value(s); in this case the command returns an empty string. *Option* may have any of the values accepted by the **tixFileEntry** command.

pathName **invoke**
: Forces the FileEntry widget to act as if the user has pressed the <return> key inside the entry subwidget.

pathName **filedialog** *?args?*
: When no additional arguments are given, this command returns the pathname of the file dialog box associated with this FileEnt widget. When additional arguments are given, the widget command of the file dialog will be called with these arguments.

pathName **subwidget** *name ?args?*
: When no options are given, this command returns the pathname of the subwidget of the specified name. When options are given, the widget command of the specified subwidget will be called with these options.

pathName **update**
: If the user has modified the entry using keyboard inputs, the update command will **update** the **−value** of this FileEntry widget. When the FileEntry widget's **−selectmode** option is set to "normal", one should call the **update** command on this widget before examining its **−value** option. This command has no effect in if the **−selectmode** option is set to "immediate".

KEYWORDS

Tix(n)

NAME

tixForm − Geometry manager based on attachment rules

SYNOPSIS

tixForm *option arg ?arg ...?*

DESCRIPTION

The **tixForm** command is used to communicate with the **tixForm** Geometry Manager, a geometry manager that arranges the geometry of the children in a parent window according to attachment rules. The **tixForm** geometry manager is very flexible and powerful; it can be used to emulate all the existing features of the Tk packer and placer geometry managers (see **pack(n)**, **place(n)**). The **tixForm** command can have any of several forms, depending on the *option* argument:

tixForm *slave* ?*options*?

If the first argument to **tixForm** is a window name (any value starting with ".''), then the command is processed in the same way as **tixForm configure**.

tixForm check *master*

This command checks whether there is circular dependency in the attachments of the master's slaves (see the section **CIRCULAR DEPENDENCY** below). It returns the Boolean value **TRUE** if it discover circular dependency and **FALSE** otherwise.

tixForm configure *slave* ?−*option value ...*?

Sets or adjusts the attachment values of the slave window according to the −*option value* argument pairs.

−b *attachment*

Abbreviation for the **−bottom** option.

−bottom *attachment*

Specifies an attachment for the bottom edge of the slave window. The attachment must specied according to the section **SPECIFYING ATTACHMENTS** below.

−bottomspring *weight*

Specifies the weight of the spring at the bottom edge of the slave window. See the section **USING SPRINGS** below.

−bp *value*

Abbreviation for the **−padbottom** option.

−bs *weight*

Abbreviation for the **−bottomspring** option.

−fill *master*

Specifies the fillings when springs are used for this widget. The value must be **x**, **y**, **both** or **none**.

−in *master*

Places the slave window into the specified master window. If the slave was originally in another master window, all attachment values with respect to the original master window are discarded. Even if the attachment values are the same as in the original master window, they need to be specified again. The **−in** flag, when needed, must appear as the first flag after the name of the slave. Otherwise an error is generated.

−l *attachment*

Abbreviation for the **−left** option.

−left *attachment*

Specifies an attachment for the left edge of the slave window. The attachment must specied according to the section **SPECIFYING ATTACHMENTS** below.

−leftspring *weight*
Specifies the weight of the spring at the left edge of the slave window. See the section **USING SPRINGS** below.

−lp *value*
Abbreviation for the **−padleft** option.

−ls *weight*
Abbreviation for the **−leftspring** option.

−padbottom *value*
Specifies the amount of external padding to leave on the bottom side of the slave. The *value* may have any of the forms acceptable to **Tk_GetPixels(3)**.

−padleft *value*
Specifies the amount of external padding to leave on the left side of the slave.

−padright *value*
Specifies the amount of external padding to leave on the right side of the slave.

−padtop *value*
Specifies the amount of external padding to leave on the top side of the slave.

−padx *value*
Specifies the amount of external padding to leave on both the left and the right sides of the slave.

−pady *value*
Specifies the amount of external padding to leave on both the top and the bottom sides of the slave.

−r *attachment*
Abbreviation for the **−right** option.

−right *attachment*
Specifies an attachment for the right edge of the slave window. The attachment must specied according to the section **SPECIFYING ATTACHMENTS** below.

−rightspring *weight*
Specifies the weight of the spring at the right edge of the slave window. See the section **USING SPRINGS** below.

−rp *value*
Abbreviation for the **−padright** option.

−rs *weight*
Abbreviation for the **−rightspring** option.

−t *attachment*
Abbreviation for the **−top** option.

−top *attachment*
Specifies an attachment for the top edge of the slave window. The attachment must specied according to the section **SPECIFYING ATTACHMENTS** below.

−topspring *weight*
Specifies the weight of the spring at the top edge of the slave window. See the section **USING SPRINGS** below.

−tp *value*
Abbreviation for the **−padtop** option.

−ts *weight*

Abbreviation for the **−topspring** option.

tixForm forget *slave* ?*slave* ...?

Removes each of the slaves from its master and unmaps their windows. The slaves will no longer be managed by tixForm. All attachment values with respect to their master windows are discarded. If another slave is attached to this slave, then the attachment of the other slave will be changed to grid attachment based on its geometry.

tixForm grid *master* ?*x_size y_size*?

When *x_size* and *y_size* are given, this command returns the number of grids of the master window in a pair of integers of the form {*x_size y_size*}. When both *x_size* and *y_size* are given, this command changes the number of horizontal and vertical grids on the master window.

tixForm info *slave* ?*option*?

Queries the attachment options of a slave window. *option* can be any of the options accepted by the **tixForm configure** command. If *option* is given, only the value of that option is returned. Otherwise, this command returns a list whose elements are the current configuration state of the slave given in the same *option−value* form that might be specified to **tixForm configure**. The first two elements in this list list are "**−in** *master*" where *master* is the slave's master window.

tixForm slaves *master*

Returns a list of all of the slaves for the master window. The order of the slaves in the list is the same as their order in the packing order. If master has no slaves then an empty string is returned.

SPECIFYING ATTACHMENTS

One can specify an attachment for each side of a slave window managed by tixForm. An attachment is specified in the the form "*−side* {*anchor_point offset*}". *−side* can be one of **−top**, **−bottom**, **−left** or **−right**.

Offset is given in screen units (i.e. any of the forms acceptable to **Tk_GetPixels**). A positive offset indicates shifting to a position to the right or bottom of an anchor point. A negative offset indicates shifting to a position to the left or top of an anchor point.

Anchor_point can be given in one of the following forms:

Grid Attachment

The master window is divided into a number of horizontal and vertical grids. By default the master window is divided into 100x100 grids; the number of grids can be adjusted by the **tixForm grid** command. A grid attachment anchor point is given by a **%** sign followed by an **integer** value. For example, **%0** spceifies the first grid line (the top or left edge of the master window). **%100** spceifies the last grid line (the bottom or right edge of the master window).

Opposite Side Attachment

Opposite attachment specifies an anchor point located on the **opposite** side of another slave widget, which must be managed by tixForm in the same master window. An opposite attachment anchor point is given by the name of another widget. For example, "tixForm .b −top {.a 0}" attaches the **top** side of the widget **.b** to the **bottom** of the widget **.a**.

Parallel Side Attachment

Opposite attachment specifies an anchor point located on the **same** side of another slave widget, which must be managed by tixForm in the same master window. An parallel attachment anchor point is given by the sign **&** follwed by the name of another widget. For example, "tixForm .b −top {&.a 0}" attaches the **top** side of the widget **.b** to the **top** of the widget **.a**, making the **top** sides of these two widgets at the same vertical position in their parent window.

No Attachment

Specifies a side of the slave to be attached to nothing, indicated by the keyword **none**. When the **none** anchor point is given, the offser must be zero.

When a side of a slave is attached to **{none 0}**, the position of this side is calculated by the position of the other side and the natural size of the slave. For example, if a the **left** side of a widget is attached to **{%0 100}**, its **right** side attached to **{none 0}**, and the natural size of the widget is **50** pixels, the **right** side of the widget will be positioned at pixel **{%0 149}**.

When both **–top** and **–bottom** are attached to **none**, then by default **–top** will be attached to **{%0 0}**. When both **–left** and **–right** are attached to none, then by default **–left** will be attached to **{%0 0}**.

Shifting effects can be achieved by specifying a non-zero offset with an anchor point. In the following example, the **top** side of widget **.b** is attached to the **bottom** of **.a**; hence **.b** always appears below **.a**. Also, the left edge of **.b** is attached to the **left** side of **.a** with a 10 pixel offest. Therefore, the **left** edge of **.b** is always shifted 10 pixels to the right of **.a**'s **left** edge:

```
tixForm .b –left {.a 10} –top {.a 0}
```

ABBREVIATIONS: Certain abbreviations can be made on the attachment specifications: First an offset of zero can be omitted. Thus, the following two lines are equivalent:

```
tixForm .b –top {.a 0} –right {%100 0}
tixForm .b –top {.a}   –right {%100}
```

Also, because of the way TCL handles lists, when you omit the offset, you can also leave out the braces. So you can further simplify the above to:

```
tixForm .b –top .a –right %100
```

In the second case, when the anchor point is omitted, the offset must be given. A default anchor point is chosen according to the value of the offset. If the anchor point is **0** or positive, the default anchor point %0 is used; thus, "tixForm .b –top 15" attaches the top edge of **.b** to a position 15 pixels below the top edge of the master window. If the anchor point is "**-0**" or negative, the default anchor point **%100** is used; thus, "tixForm .a –right –2" attaches the right edge of **.a** to a position 2 pixels to the left of the master window's **right** edge. An further example below shows a command with its equivalent abbreviation.

```
tixForm .b –top {%0 10} –bottom {%100 0}
tixForm .b –top 10      –bottom –0
```

USING SPRINGS

To be written.

ALGORITHM OF TIXFORM

TixForm starts with any slave in the list of slaves of the master window. Then it tries to determine the position of each side of the slave.

If the attachment of a side of the slave is grid attachment, the position of the side is readily determined.

If the attachment of this side is **none**, then tixForm tries to determine the position of the opposite side first, and then use the position of the opposite side and the natural size of the slave to determine the position of this side.

If the attachment is opposite or parallel widget attachments, then tixForm tries to determine the positions of the other widget first, and then use the positions of the other widget and the natural size of the slave determine the position of this side. This recursive algorithmis carried on until the positions of all slaves are determined.

CIRCULAR DEPENDENCY

The algorithm of tixForm will fail if a circular dependency exists in the attachments of the slaves. For example:

```
tixForm .c –left .b
tixForm .b –right .c
```

In this example, the position of the left side of **.b** depends on the right side of **.c**, which in turn depends on the left side of **.b**.

When a circular dependency is discovered during the execution of the tixForm algorithm, tixForm will generate a background error and the geometry of the slaves are undefined (and will be arbitrary). Notice that tixForm only executes the algorithm when the specification of the slaves' attachments is complete. Therefore, it allows intermediate states of circular dependency during the specification of the slaves' attachments. Also, unlike the Motif Form manager widget, tixForm defines circular dependency as "*dependency in the same dimension*". Therefore, the following code fragment will does not have circular dependency because the two widgets do not depend on each other in the same dimension (**.b** depends **.c** in the horizontal dimension and **.c** depends on **.b** in the vertical dimension):

```
tixForm .b –left .c
tixForm .c –top .b
```

BUGS

Springs have not been fully implemented yet.

KEYWORDS

Tix(n), Form, Geometry Management

NAME

tixGetBoolean - Get the boolean value of a string.

SYNOPSIS

tixGetBoolean *?-nocomplain? string*

DESCRIPTION

The command **tixGetBoolean** returns "0" if the string is a valid TCL string for the boolean value FALSE. It returns "1" if the string is a valid TCL string for the boolean value TRUE.

When the string is not a valid TCL boolean value and the **-nocomplain** option is specified, **tixGetBoolean** will return "0". Otherwise it will generate an error.

KEYWORDS

Tix(n)

NAME

tixGetInt - Get the integer value of a string.

SYNOPSIS

tixGetInt *?-nocomplain? ?-trunc? string*

DESCRIPTION

The command **tixGetInt** converts any number into an integer number. By default, it will round the number to the nearest integer. When the **-trunc** option is specified, the number is truncated instead of rounded.

When the string is not a valid TCL numerical value and the **-nocomplain** option is specified, **tixGetInt** will return "0". Otherwise it will generate an error.

KEYWORDS

Tix(n)

NAME

tixHList − Create and manipulate Tix Hierarchial List widgets

SYNOPSIS

tixHList *pathName ?options*?

SUPER-CLASS

None.

STANDARD OPTIONS

background	**borderWidth**	**cursor**	**foreground**
font	**height**	**highlightColor**	**highlightThickness**
relief	**selectBackground**	**selectForeground**	
xScrollCommand	**yScrollCommand**	**width**	

See the **options(n)** manual entry for details on the standard options.

WIDGET-SPECIFIC OPTIONS

Name: **browsecmd**
Class: **BrowseCmd**
Switch: **−browsecmd**

Specifies a TCL command to be executed when the user browses through the entries in the HList widget.

Name: **columns**
Class: **Columns**
Switch: **−columns**

Specifies the number of columns in this HList widget. This option can only be set during the creation of the HList widget and cannot be changed subsequently.

Name: **command**
Class: **Command**
Switch: **−command**

Specifies the TCL command to be executed when the user invokes a list entry in the HList widget. Normally the user invokes a list entry by double-clicking it or pressing the Return key.

Name: **drawBranch**
Class: **DrawBranch**
Switch: **−drawbranch**

A Boolean value to specify whether branch line should be drawn to connect list entries to their parents.

Name: **foreground**
Class: **Foreground**
Switch: **−foreground**
Alias: **−fg**

[OBSOLETE] Specifies the default foreground color for the list entries.

Name: **gap**
Class: **Gap**
Switch: **−gap**

[OBSOLETE] The default distance between the bitmap/image and the text in list entries.

Name: **header**
Class: **Header**
Switch: **−header**

A Boolean value specifying whether headers should be displayed for this HList widget (see the **header** widget command below).

Name: **height**
Class: **Height**
Switch: **–height**

Specifies the desired height for the window in number of characters.

Name: **indent**
Class: **Indent**
Switch: **–indent**

Specifies the amount of horizontal indentation between a list entry and its children. Must be a valid screen distance value.

Name: **indicator**
Class: **Indicator**
Switch: **–indicator**

Specifies whether the indicators should be displayed inside the HList widget. See the **indicator** widget command below.

Name: **indicatorCmd**
Class: **IndicatorCmd**
Switch: **–indicatorcmd**

Specifies a TCL command to be executed when the user manipulates the indicator of an HList entry. The **–indicatorcmd** is triggered when the user press or releases the mouse button over the indicator in an HList entry. By default the TCL command specified by **–indicatorcmd** is executed with one additional argument, the entryPath of the entry whose indicator has been triggered. Additional information about the event can be obtained by the **tixEvent** command.

Name: **itemType**
Class: **ItemType**
Switch: **–itemtype**

Specifies the default type of display item for this HList widget. When you call the add and addchild widget commands, display items of this type will be created if the **–itemtype** option is not specified .

Name: **padX**
Class: **Pad**
Switch: **–padx**

[OBSOLETE] The default horizontal padding for list entries.

Name: **padY**
Class: **Pad**
Switch: **–padx**

[OBSOLETE] The default vertical padding for list entries.

Name: **selectBackground**
Class: **SelectBackground**
Switch: **–selectbackground**

Specifies the background color for the selected list entries.

Name: **selectBorderWidth**
Class: **BorderWidth**
Switch: **–selectborderwidth**

Specifies a non-negative value indicating the width of the 3-D border to draw around selected items. The value may have any of the forms acceptable to **Tk_GetPixels**.

Name: **selectForeground**
Class: **SelectForeground**
Switch: **−selectforeground**

Specifies the foreground color for the selected list entries.

Name: **selectMode**
Class: **SelectMode**
Switch: **−selectmode**

Specifies one of several styles for manipulating the selection. The value of the option may be arbitrary, but the default bindings expect it to be either **single**, **browse**, **multiple**, or **extended**; the default value is **single**.

Name: **sizeCmd**
Class: **SizeCmd**
Switch: **−sizecmd**

Specifies a TCL script to be called whenever the HList widget changes its size. This command can be useful to implement "user scroll bars when needed" features.

Name: **separator**
Class: **Separator**
Switch: **−separator**

Specifies the character to used as the separator character when intepreting the path-names of list entries. By default the character "." is used.

Name: **width**
Class: **Width**
Switch: **−width**

Specifies the desired width for the window in characters.

DESCRIPTION

The **tixHList** command creates a new window (given by the *pathName* argument) and makes it into a HList widget. Additional options, described above, may be specified on the command line or in the option database to configure aspects of the HList widget such as its cursor and relief.

The HList widget can be used to display any data that have a hierarchical structure, for example, file system directory trees. The list entries are indented and connected by branch lines according to their places in the hierachy.

Each list entry is identified by an **entryPath**. The entryPath is a sequence of **entry names** separated by the separator charactor (specified by the **−separator** option). An **entry name** can be any string that does not contain the separator charactor, or it can be the a string that contains only one separator charactor.

For example, when "." is used as the separator charactor, "one.two.three" is the entryPath for a list entry whose parent is "one.two", whose parent is "one", which is a toplevel entry (has no parents).

Another examples: ".two.three" is the entryPath for a list entry whose parent is ".two", whose parent is ".", which is a toplevel entry.

DISPLAY ITEMS

Each list entry in an HList widget is associated with a **display item**. The display item determines what visual information should be displayed for this list entry. Please see the **DItem(n)** manual page for a list of all display items.

When a list entry is created by the **add** or **addchild** widget commands, the type of its display item is determined by the **–itemtype** option passed to these commands. If the **–itemtype** is omitted, then by default the type specified by this HList widget's **–itemtype** option is used.

WIDGET COMMAND

The **tixHList** command creates a new Tcl command whose name is the same as the path name of the HList widget's window. This command may be used to invoke various operations on the widget. It has the following general form:

pathName option ?*arg arg ...*?

PathName is the name of the command, which is the same as the HList widget's path name. *Option* and the *arg*s determine the exact behavior of the command. The following commands are possible for HList widgets:

pathName **add** *entryPath* ?*option value ...*?

Creates a new list entry with the pathname *entryPath*. A list entry must be created after its parent is created (unless this entry is a top-level entry, which has no parent). This command returns the entryPath of the newly created list entry. The following configuration options can be given to configure the list entry:

–at *position*

Insert the new list at the position given by *position*. *position* must be a valid integer. the Position **0** indicates the first position, **1** indicates the second position, and so on.

–after *afterWhich*

Insert the new list entry after the entry identified by *afterWhich*. *afterWhich* must be a valid list entry and it mush have the same parent as the new list entry

–before *beforeWhich*

Insert the new list entry before the entry identified by *beforeWhich*. *beforeWhich* must be a valid list entry and it mush have the same parent as the new list entry

–data *string*

Specifies a string to associate with this list entry. This string can be queried by the **info** widget command. The application programmer can use the **–data** option to associate the list entry with the data it represents.

–itemtype *type*

Specifies the type of display item to be display for the new list entry. **type** must be a valid display item type. Currently the available display item types are **imagetext**, **text**, and **window**. If this option is not specified, then by default the type specified by this HList widget's **–itemtype** option is used.

–state Specifies whether this entry can be selected or invoked by the user. Must be either **normal** or **disabled**.

The **add** widget command accepts additional configuration options to configure the display item associated with this list entry. The set of additional configuration options depends on the type of the display item given by the **–itemtype** option. Please see the **DItem(n)** manual page for a list of the configuration options for each of the display item types.

pathName **addchild** *parentPath* ?*option value ...* ?

Adds a new child entry to the children list of the list entry identified by *parentPath*. Or, if *parentPath* is set to be the empty string, then creates a new toplevel entry. The name of the new list entry will be a unique name automatically generated by the HList widget. Usually if *parentPath* is **foo**, then the entryPath of the new entry will be **foo.1**, **foo.2**, ... etc. This command returns the entryPath of the newly created list entry. *option* can be any option for the **add** widget command.

pathName **anchor set** *entryPath*

Sets the anchor to the list entry identified by *entryPath*. The anchor is the end of the selection that is fixed while dragging out a selection with the mouse.

pathName **anchor clear**

Removes the anchor, if any, from this HList widget. This only removes the surrounding highlights of the anchor entry and does not affect its selection status.

pathName **cget** *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the **tixHList** command.

pathName **column width** *col* ?−*char*? ?*width*?

Querys or sets the width of a the column *col* in the HList widget. The value of *col* is zero-based: 0 stands for the first column, 1 stands for the second, and so on. If no further parameters are given, returns the current width of this column (in number of pixels). Additional parameters can be given to set the width of this column:

pathName **column width** *col* **{}**

An empty string indicates that the width of the column should be just wide enough to display the widest element in this column. In this case, the width of this column may change as a result of the elements in this column changing their sizes.

pathName **column width** *col width*

width must be in a form accepted by **Tk_GetPixels(3)**.

pathName **column width** *col* **-char** *nChars*

The width is set to be the average width occupied by *nChars* number of characters of the font specified by the **−font** option of this HList widget.

pathName **configure** ?*option*? *?value option value ...*?

Query or modify the configuration options of the widget. If no *option* is specified, returns a list describing all of the available options for *pathName* (see **Tk_ConfigureInfo** for information on the format of this list). If *option* is specified with no *value*, then the command returns a list describing the one named option (this list will be identical to the corresponding sublist of the value returned if no *option* is specified). If one or more *option−value* pairs are specified, then the command modifies the given widget option(s) to have the given value(s); in this case the command returns an empty string. *Option* may have any of the values accepted by the **tixHList** command.

pathName **delete** *option* ?*entryPath*?

Delete one or more list entries. *option* may be one of the following:

all Delete all entries in the HList. In this case the *entryPath* does not need to be specified.

entry Delete the entry specified by *entryPath* and all its offsprings, if any.

offsprings

Delete all the offsprings, if any, of the entry specified by *entryPath*. However, *entryPath* itself is not deleted.

siblings

Delete all the list entries that share the same parent with the entry specified by *entryPath*. However, *entryPath* itself is not deleted.

pathName **dragsite set** *entryPath*

Sets the dragsite to the list entry identified by *entryPath*. The dragsite is used to indicate the source of a drag-and-drop action. Currently drag-and-drop functionality has not been implemented in Tix yet.

pathName **dragsite clear**

Remove the dragsite, if any, from the this HList widget. This only removes the surrounding highlights of the dragsite entry and does not affect its selection status.

pathName **dropsite set** *entryPath*

Sets the dropsite to the list entry identified by *entryPath*. The dropsite is used to indicate the target of a grag-and-drop action. Currently drag-and-drop functionality has not been implemented in Tix yet.

pathName **dropsite clear**

Remove the dropsite, if any, from the this HList widget. This only removes the surrounding highlights of the dropsite entry and does not affect its selection status.

pathName **entrycget** *entryPath option*

Returns the current value of the configuration option given by *option* for the entry indentfied by *entryPath*. *Option* may have any of the values accepted by the **add** widget command.

pathName **entryconfigure** *entryPath* ?*option*? *?value option value ...*?

Query or modify the configuration options of the list entry indentfied by *entryPath*. If no *option* is specified, returns a list describing all of the available options for *entryPath* (see **Tk_ConfigureInfo** for information on the format of this list). If *option* is specified with no *value*, then the command returns a list describing the one named option (this list will be identical to the corresponding sublist of the value returned if no *option* is specified). If one or more *option−value* pairs are specified, then the command modifies the given option(s) to have the given value(s); in this case the command returns an empty string. *Option* may have any of the values accepted by the **add** or **addchild** widget command. The exact set of options depends on the value of the **−itemtype** option passed to the the **add** or **addchild** widget command when this list entry is created.

pathName **header** *option col* ?*args ...*?

Manipulates the header items of this HList widget. If the **−header** option of this HList widget is set to true, then a header item is displayed at the top of each column. The *col* argument for this command must be a valid integer. 0 indicates the first column, 1 the second column, ... and so on. This command supports the following options:

pathName **header cget** *col option*

If the *col*-th column has a header display item, returns the value of the specified *option* of the header item. If the header doesn't exist, returns an error.

pathName **header configure** *col* ?*option*? *?value option value ...*?

Query or modify the configuration options of the header display item of the *col*-th column. The header item must exist, or an error will result. If no *option* is specified, returns a list describing all of the available options for the header display item (see **Tk_ConfigureInfo(3)** for information on the format of this list). If *option* is specified with no *value*, then the command returns a list describing the one named option (this list will be identical to the corresponding sublist of the value returned if no *option* is specified). If one or more *option−value* pairs are specified, then the command modifies the given option(s) to have the given value(s); in this case the command returns an empty string. *Option* may have any of the values accepted by the **header create** widget command. The exact set of options depends on the value of the **−itemtype** option passed to the the **header create** widget command when this display item was created.

pathName **header create** *col* ?*−itemtype type*? ?*option value ...*?

Creates a new display item as the header for the *col*-th column. If an header display item already exists for this column, it will be replaced by the new item. An optional parameter *−itemtype* can be used to specify what type of display item should be created. If the *−itemtype* is not given, then by default the type specified by this HList widget's **−itemtype** option is used. Additional parameters, in *option-value* pairs, can be passed to configure the appearance of the display item. Each *option-value* pair must be a valid option for this type of display item or one of the following:

−borderwidth
Specifies the border width of this header item.

−headerbackground
Specifies the background color of this header item.

−relief Specifies the relief type of the border of this header item.

pathName **header delete** *col*
Deletes the header display item for the *col*-th column.

pathName **header exists** *col*
Return true if an header display item exists for the *col*-th column; return false otherwise.

pathName **header size** *entryPath*
If an header display item exists for the *col*-th column , returns its size in a two element list of the form {*width height*}; returns an error if the header display item does not exist.

pathName **hide** *option ?entryPath?*
Makes some of entries invisible invisible without deleting them. *Option* can be one of the following:

entry Hides the list entry identified by *entryPath*.

Currently only the **entry** option is supported. Other options will be added in the next release.

pathName **indicator** *option* entryPath ?*args ...*?
Manipulates the indicator on the list entries. An indicator is usually a small display item (such as an image) that is displayed to the left to an entry to indicate the status of the entry. For example, it may be used to indicator whether a directory is opened or closed. *option* can be one of the following:

pathName **indicator cget** *entryPath option*
If the list entry given by *entryPath* has an indicator, returns the value of the specified *option* of the indicator. If the indicator doesn't exist, returns an error.

pathName **indicator configure** *entryPath* ?*option*? *?value option value* ...?
Query or modify the configuration options of the indicator display item of the entry specified by *entryPath*. The indicator item must exist, or an error will result. If no *option* is specified, returns a list describing all of the available options for the indicator display item (see **Tk_ConfigureInfo(3)** for information on the format of this list). If *option* is specified with no *value*, then the command returns a list describing the one named option (this list will be identical to the corresponding sublist of the value returned if no *option* is specified). If one or more *option−value* pairs are specified, then the command modifies the given option(s) to have the given value(s); in this case the command returns an empty string. *Option* may have any of the values accepted by the **indicator create** widget command. The exact set of options depends on the value of the **−itemtype** option passed to the the **indicator create** widget command when this display item was created.

pathName **indicator create** *entryPath* ?−*itemtype type*? ?*option value ...*?
Creates a new display item as the indicator for the entry specified by *entryPath*. If an indicator display item already exists for this entry, it will be replaced by the new item. An optional parameter −*itemtype* can be used to specify what type of display item should be created. If the −*itemtype* is not given, then by default the type specified by this HList widget's **−itemtype** option is used. Additional parameters, in *option-value* pairs, can be passed to configure the appearance of the display item. Each *option-value* pair must be a valid option for this type of display item.

pathName **indicator delete** *entryPath*

Deletes the indicator display item for the entry given by *entryPath*.

pathName **indicator exists** *entryPath*

Return true if an indicator display item exists for the entry given by *entryPath*; return false otherwise.

pathName **indicator size** *entryPath*

If an indicator display item exists for the entry given by *entryPath*, returns its size in a two element list of the form {*width height*}; returns an error if the indicator display item does not exist.

pathName **info** *option arg ...*

Query information about the HList widget. *option* can be one of the following:

pathName **info anchor** *entryPath*

; Returns the entryPath of the current anchor, if any, of the HList widget. If the anchor is not set, returns the empty string.

pathName **info children** ?*entryPath*?

If *entrpyPath* is given, returns a list of the entryPath's of its children entries. Otherwise returns a list of the toplevel entryPath's.

pathName **info data** ?*entryPath*?

Returns the data associated with *entryPath*.

pathName **info dragsite** *entryPath*

Returns the entryPath of the current dragsite, if any, of the HList widget. If the dragsite is not set, returns the empty string.

pathName **info dropsite** *entryPath*

Returns the entryPath of the current dropsite, if any, of the HList widget. If the dropsite is not set, returns the empty string.

pathName **info exists** *entryPath*

Returns a boolean value indicating whether the list entry *entrpyPath* exists.

pathName **info hidden** *entryPath*

Returns a boolean value indicating whether the list entry **entrpyPath** is hidden or not.

pathName **info next** *entryPath*

Returns the entryPath of the list entry, if any, immediately below this list entry. If this entry is already at the bottom of the HList widget, returns an empty string.

pathName **info parent** *entryPath*

Returns the name of the parent of the list entry identified by *entrpyPath*. If *entrpyPath* is a toplevel list entry, returns the empty string.

pathName **info prev** *entryPath*

Returns the entryPath of the list entry, if any, immediately above this list entry. If this entry is already at the top of the HList widget, returns an empty string.

pathName **info selection**

Returns a list of selected elements in the HList widget. If no entries are selectd, returns an empty string.

pathName **item** *option* ?*args ...*?

Creates and configures the display items at individual columns the entries. The form of additional of arguments depends on the choice of *option*:

pathName **item cget** *entryPath col option*
Returns the current value of the configure *option* of the display item at the column designated by *col* of the entry specified by *entryPath*.

pathName **item configure** *entryPath col* ?*option*? ?*value option value* ...?
Query or modify the configuration options of the display item at the column designated by *col* of the entry specified by *entryPath*. If no *option* is specified, returns a list describing all of the available options for *entryPath* (see **Tk_ConfigureInfo(3)** for information on the format of this list). If *option* is specified with no *value*, then the command returns a list describing the one named option (this list will be identical to the corresponding sublist of the value returned if no *option* is specified). If one or more *option−value* pairs are specified, then the command modifies the given option(s) to have the given value(s); in this case the command returns an empty string. *Option* may have any of the values accepted by the **item create** widget command. The exact set of options depends on the value of the **−itemtype** option passed to the the **item create** widget command when this display item was created.

pathName **item create** *entryPath col* ?*−itemtype type*? ?*option value* ...?
Creates a new display item at the column designated by *col* of the entry specified by *entryPath*. An optional parameter *−itemtype* can be used to specify what type of display items should be created. If the *−itemtype* is not specified, then by default the type specified by this HList widget's **−itemtype** option is used. Additional parameters, in *option-value* pairs, can be passed to configure the appearance of the display item. Each *option-value* pair must be a valid option for this type of display item.

pathName **item delete** *entryPath col*
Deletes the display item at the column designated by *col* of the entry specified by *entryPath*.

pathName **item exists** *entryPath col*
Returns true if there is a display item at the column designated by *col* of the entry specified by *entryPath*; returns false otherwise.

pathName **nearest** *y*
Given a y-coordinate within the HList window, this command returns the entryPath of the (visible) HList element nearest to that y-coordinate.

pathName **see** *entryPath*
Adjust the view in the HList so that the entry given by *entryPath* is visible. If the entry is already visible then the command has no effect; if the entry is near one edge of the window then the HList scrolls to bring the element into view at the edge; otherwise the HList widget scrolls to center the entry.

pathName **selection** *option arg* ...
This command is used to adjust the selection within a HList widget. It has several forms, depending on *option*:

pathName **selection clear** ?*from*? ?*to*?
When no extra arguments are given, deselects all of the list entrie(s) in this HList widget. When only *from* is given, only the list entry identified by *from* is deselected. When both *from* and *to* are given, deselects all of the list entrie(s) between between *from* and *to*, inclusive, without affecting the selection state of entries outside that range.

pathName **selection includes** *entryPath*
Returns 1 if the list entry indicated by *entryPath* is currently selected; returns 0 otherwise.

pathName **selection set** *from* ?*to*?
Selects all of the list entrie(s) between between *from* and *to*, inclusive, without affecting the selection state of entries outside that range. When only *from* is given, only the list

entry identified by *from* is selected.

pathName **show** *option ?entryPath?*

Show the entries that are hidden by the **hide** command, *option* can be one of the following:

entry Shows the list entry identified by *entryPath*.

Currently only the **entry** option is supported. Other options will be added in future releases.

pathName **xview** *args*

This command is used to query and change the horizontal position of the information in the widget's window. It can take any of the following forms:

pathName **xview**

Returns a list containing two elements. Each element is a real fraction between 0 and 1; together they describe the horizontal span that is visible in the window. For example, if the first element is off-screen to the left, the middle 40% is visible in the window, and 40% of the entry is off-screen to the right. These are the same values passed to scrollbars via the **−xscrollcommand** option.

pathName **xview** *entryPath*

Adjusts the view in the window so that the list entry identified by *entryPath* is aligned to the left edge of the window.

pathName **xview moveto** *fraction*

Adjusts the view in the window so that *fraction* of the total width of the HList is off-screen to the left. *fraction* must be a fraction between 0 and 1.

pathName **xview scroll** *number what*

This command shifts the view in the window left or right according to *number* and *what*. *Number* must be an integer. *What* must be either **units** or **pages** or an abbreviation of one of these. If *what* is **units**, the view adjusts left or right by *number* character units (the width of the **0** character) on the display; if it is **pages** then the view adjusts by *number* screenfuls. If *number* is negative then characters farther to the left become visible; if it is positive then characters farther to the right become visible.

pathName **yview** *?args*?

This command is used to query and change the vertical position of the entries in the widget's window. It can take any of the following forms:

pathName **yview**

Returns a list containing two elements, both of which are real fractions between 0 and 1. The first element gives the position of the list element at the top of the window, relative to the HList as a whole (0.5 means it is halfway through the HList, for example). The second element gives the position of the list entry just after the last one in the window, relative to the HList as a whole. These are the same values passed to scrollbars via the **−yscrollcommand** option.

pathName **yview** *entryPath*

Adjusts the view in the window so that the list entry given by *entryPath* is displayed at the top of the window.

pathName **yview moveto** *fraction*

Adjusts the view in the window so that the list entry given by *fraction* appears at the top of the window. *Fraction* is a fraction between 0 and 1; 0 indicates the first entry in the HList, 0.33 indicates the entry one-third the way through the HList, and so on.

pathName **yview scroll** *number what*

This command adjust the view in the window up or down according to *number* and *what*. *Number* must be an integer. *What* must be either **units** or **pages**. If *what* is **units**, the view adjusts up or down by *number* lines; if it is **pages** then the view adjusts by *number* screenfuls. If *number* is negative then earlier entries become visible; if it is positive then later entries become visible.

BINDINGS

[1] If the **−selectmode** is "browse", when the user drags the mouse pointer over the list entries, the entry under the pointer will be highlighted and the **−browsecmd** procedure will be called with one parameter, the entryPath of the highlighted entry. Only one entry can be highlighted at a time. The **−command** procedure will be called when the user double-clicks on a list entry.

[2] If the **−selectmode** is "single", the entries will only be highlighted by mouse <ButtonRelease-1> events. When a new list entry is highlighted, the **−browsecmd** procedure will be called with one parameter indicating the highlighted list entry. The **−command** procedure will be called when the user double-clicks on a list entry.

[3] If the **−selectmode** is "multiple", when the user drags the mouse pointer over the list entries, all the entries under the pointer will be highlighted. However, only a contiguous region of list entries can be selected. When the highlighted area is changed, the **−browsecmd** procedure will be called with an undefined parameter. It is the responsibility of the **−browsecmd** procedure to find out the exact highlighted selection in the HList. The **−command** procedure will be called when the user double-clicks on a list entry.

[4] If the **−selectmode** is "extended", when the user drags the mouse pointer over the list entries, all the entries under the pointer will be highlighted. The user can also make disjointed selections using <Control-ButtonPress-1>. When the highlighted area is changed, the **−browsecmd** procedure will be called with an undefined parameter. It is the responsibility of the **−browsecmd** procedure to find out the exact highlighted selection in the HList. The **−command** procedure will be called when the user double-clicks on a list entry.

[5] **Arrow key bindings:** <Up> arrow key moves the anchor point to the item right on top of the current anchor item. <Down> arrow key moves the anchor point to the item right below the current anchor item. <Left> arrow key moves the anchor to the parent item of the current anchor item. <Right> moves the anchor to the first child of the current anchor item. If the current anchor item does not have any children, moves the anchor to the item right below the current anchor item.

EXAMPLE

This example demonstrates how to use an HList to store a file directory structure and respond to the user's browse events:

```
tixHList .h -separator "/" -browsecmd browse -selectmode single \
   -itemtype text
.h add /         -text /
.h add /home     -text /home
.h add /home/ioi -text /home/ioi
.h add /home/foo -text /home/foo
.h add /usr      -text /usr
.h add /usr/lib  -text /usr/lib
pack .h

proc browse {file} {
    puts "$file browsed"
}
```

BUGS

The fact that the display item at column 0 is implicitly associated with the whole entry is probably a design bug. This was done for backward compatibility purposes. The result is that there is a large overlap between the **item** command and the **add**, **addchild**, **entrycget** and **entryconfigure** commands. Whenever multiple columns exist, the programmer should use ONLY the **item** command to create and configure the display items in each column; the **add**, **addchild**, **entrycget** and **entryconfigure** should be used ONLY to create and configure entries.

KEYWORDS

Tix(n), Hierarchical Listbox

NAME
tixInputOnly − Create and manipulate TIX **InputOnly** widgets

SYNOPSIS
tixInputOnly *pathName ?options*?

SUPER-CLASS
None

STANDARD OPTIONS
Only the following three standard options are supported by **TixInputOnly**:

cursor **width** **height**

See the "options(n)" manual entry for details on the standard options.

WIDGET-SPECIFIC OPTIONS
TixInputOnly does not have any widget specific options.

DESCRIPTION
The **tixInputOnly** command creates a new window (given by the *pathName* argument) and makes it into a **tixInputOnly** widget. Additional options, described above, may be specified on the command line or in the option database to configure aspects of the **tixInputOnly** such as its cursor or width.

TixInputOnly widgets are not visible to the user. The only purpose of **TixInputOnly** widgets are to accept inputs from the user, which can be done with the **bind** command.

WIDGET COMMAND
The **tixInputOnly** command creates a new Tcl command whose name is the same as the path name of the **tixInputOnly**'s window. This command may be used to invoke various operations on the widget. It has the following general form:

pathName option ?*arg arg ...*?

PathName is the name of the command, which is the same as the InputOnly widget's path name. *Option* and the *arg*s determine the exact behavior of the command. The following commands are possible for **tixInputOnly** widgets:

pathName **cget** *option*
: Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the **tixInputOnly** command.

pathName **configure** ?*option*? *?value option value ...*?
: Query or modify the configuration options of the widget. If no *option* is specified, returns a list describing all of the available options for *pathName* (see **Tk_ConfigureInfo** for information on the format of this list). If *option* is specified with no *value*, then the command returns a list describing the one named option (this list will be identical to the corresponding sublist of the value returned if no *option* is specified). If one or more *option−value* pairs are specified, then the command modifies the given widget option(s) to have the given value(s); in this case the command returns an empty string. *Option* may have any of the values accepted by the **tixInputOnly** command.

BINDINGS
tixInputOnly widgets have no default bindings.

KEYWORDS

Tix(n)

NAME

tixLabelEntry − Create and manipulate tixLabelEntry widgets

SYNOPSIS

tixLabelEntry *pathName ?options*?

SUPER-CLASS

The **TixLabelEntry** class is derived from the **TixLabelWidget** class and inherits all the commands, options and subwidgets of its super-class.

STANDARD OPTIONS

The LabelEntry widget supports all the standard options of a frame widget. See the **options(n)** manual entry for details on the standard options.

WIDGET-SPECIFIC OPTIONS

Name: **disableForeground**
Class: **DisableForeground**
Switch: **−disableforeground**

The foreground color to use for of the entry subwidget when the LabelEntry widget is disabled.

Name: **label**
Class: **Label**
Switch: **−label**

Specifies the string to display as the label of this LabelEntry widget.

Name: **labelSide**
Class: **LabelSide**
Switch: **−labelside**

Specifies where the label should be displayed relative to the entry subwidget. Valid options are: **top**, **left**, **right**, **bottom**, **none** or **acrosstop**.

Name: **state**
Class: **State**
Switch: **−state**

Specifies the whether the LabelEntry widget is normal or disabled. Only the values "normal" and "disabled" are recognized.

SUBWIDGETS

Name: **label**
Class: **Label**

The label subwidget.

Name: **entry**
Class: **Entry**

The entry subwidget.

DESCRIPTION

The **tixLabelEntry** command creates a new window (given by the *pathName* argument) and makes it into a LabelEntry widget. Additional options, described above, may be specified on the command line or in the option database to configure aspects of the LabelEntry such as its cursor and relief.

The LabelEntry widget packages an entry widget and a label into one mega widget. It can be used be used to simplify the creation of "entry-form" type of interface. In this kind of interface, one must create many entry widgets with label widgets next to them and describe the use of each of the entry widgets.

WIDGET COMMANDS

The **tixLabelEntry** command creates a new Tcl command whose name is the same as the path name of the LabelEntry's window. This command may be used to invoke various operations on the widget. It has the following general form:

pathName option ?*arg arg ...*?

PathName is the name of the command, which is the same as the LabelEntry widget's path name. *Option* and the *arg*s determine the exact behavior of the command. The following commands are possible for LabelEntry widgets:

pathName **cget** *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the **tixLabelEntry** command.

pathName **configure** ?*option*? *?value option value ...*?

Query or modify the configuration options of the widget. If no *option* is specified, returns a list describing all of the available options for *pathName* (see **Tk_ConfigureInfo** for information on the format of this list). If *option* is specified with no *value*, then the command returns a list describing the one named option (this list will be identical to the corresponding sublist of the value returned if no *option* is specified). If one or more *option−value* pairs are specified, then the command modifies the given widget option(s) to have the given value(s); in this case the command returns an empty string. *Option* may have any of the values accepted by the **tixLabelEntry** command.

pathName **subwidget** *name ?args?*

When no options are given, this command returns the pathname of the subwidget of the specified name.

When options are given, the widget command of the specified subwidget will be called with these options.

KEYWORDS

Tix(n)

NAME

tixLabelFrame − Create and manipulate tixLabelFrame widgets

SYNOPSIS

tixLabelFrame *pathName ?options*?

SUPER-CLASS

The **TixLabelFrame** class is derived from the **TixLabelWidget** class and inherits all the commands, options and subwidgets of its super-class.

STANDARD OPTIONS

The LabelFrame widget supports all the standard options of a frame widget. See the **options(n)** manual entry for details on the standard options.

WIDGET-SPECIFIC OPTIONS

Name: **label**
Class: **Label**
Switch: **−label**

Specifies the string to display as the label of this LabelFrame widget.

Name: **labelSide**
Class: **LabelSide**
Switch: **−labelside**

Specifies where the label should be displayed relative to the entry subwidget. Valid options are: **top**, **left**, **right**, **bottom**, **none** or **acrosstop**.

Name: **padX**
Class: **Pad**
Switch: **−padx**

Specifies the amount of the horizontal padding around the **frame** subwidget. Must be a valid non-negative integer number.

Name: **padY**
Class: **Pad**
Switch: **−pady**

Specifies the amount of the vertical padding around the **frame** subwidget.

SUBWIDGETS

Name: **frame**
Class: **Frame**

The frame subwidget.

Name: **label**
Class: **Label**

The label subwidget.

DESCRIPTION

The **tixLabelFrame** command creates a new window (given by the *pathName* argument) and makes it into a LabelFrame widget. Additional options, described above, may be specified on the command line or in the option database to configure aspects of the LabelFrame such as its cursor and relief.

CREATING WIDGETS INSIDE A LABELFRAME

The LabelFrame widget packages a frame widget and a label into one mega widget. To create widgets inside a LabelFrame widget, one must create the new widgets relative to the **frame** subwidget and manage them inside the **frame** subwidget. An error will be generated if one tries to create widgets as immediate children of the LabelFrame. For example: the following is correct code, which creates new widgets inside the frame subwidget:

```
tixLabelFrame .f
set f [.f subwidget frame]
button $f.b −text hi
pack $f.b
```

The following example code is *incorrect* because it tries to create immediate children of the LabelFrame **.f**:

```
tixLabelFrame .f
button .f.b −text hi
pack .f.b
```

WIDGET COMMANDS

The **tixLabelFrame** command creates a new Tcl command whose name is the same as the path name of the LabelFrame's window. This command may be used to invoke various operations on the widget. It has the following general form:

pathName option ?*arg arg ...*?

PathName is the name of the command, which is the same as the LabelFrame widget's path name. *Option* and the *arg*s determine the exact behavior of the command. The following commands are possible for LabelFrame widgets:

pathName **cget** *option*
: Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the **tixLabelFrame** command.

pathName **configure** ?*option*? *?value option value ...*?
: Query or modify the configuration options of the widget. If no *option* is specified, returns a list describing all of the available options for *pathName* (see **Tk_ConfigureInfo** for information on the format of this list). If *option* is specified with no *value*, then the command returns a list describing the one named option (this list will be identical to the corresponding sublist of the value returned if no *option* is specified). If one or more *option−value* pairs are specified, then the command modifies the given widget option(s) to have the given value(s); in this case the command returns an empty string. *Option* may have any of the values accepted by the **tixLabelFrame** command.

pathName **subwidget** *name ?args?*
: When no options are given, this command returns the pathname of the subwidget of the specified name.

: When options are given, the widget command of the specified subwidget will be called with these options.

KEYWORDS

Tix(n)

NAME

tixListNoteBook - Create and manipulate tixListNoteBook widgets

SYNOPSIS

tixListNoteBook *pathName ?options*?

STANDARD OPTIONS

The ListNoteBook widget supports all the standard options of a frame widget. See the options(n) manual entry for details on the standard options.

WIDGET-SPECIFIC OPTIONS

Name: **dynamicGeometry**
Class: **DynamicGeometry**
Switch: **–dynamicgeometry**

If set to false, the size of the ListNotebook will match the size of the largest page. If set to true, the size of the ListNotebook will match the size of the current page (therefore, the size may change when the user selects different pages). The default value is false. A setting of true is discouraged.

Name: **ipadX**
Class: **Pad**
Switch: **–ipadx**

The amount of internal horizontal paddings around the sides of the page subwidgets.

Name: **ipadY**
Class: **Pad**
Switch: **–ipady**

The amount of internal vertical paddings around the sides of the page subwidgets.

SUBWIDGETS

Name: **hlist**
Class: **TixHList**

The HList widget that displays the names of the pages.

In addition, all the page subwidgets created as a result of the **add** command can be accessed by the **subwidget** command. They are identified by the **pageName** parameter to the **add** command.

DESCRIPTION

The **tixListNoteBook** command creates a new window (given by the *pathName* argument) and makes it into a ListNoteBook widget. Additional options, described above, may be specified on the command line or in the option database to configure aspects of the ListNoteBook widget such as its cursor and relief.

The ListNoteBook widget is very similar to the TixNoteBook widget: it can be used to display many windows in a limited space using a "notebook" metaphore. The notebook is divided into a stack of pages (windows). At one time only one of these pages can be shown. The user can navigate through these pages by choosing the name of the desired page in the **hlist** subwidget.

WIDGET COMMANDS

The **tixListNoteBook** command creates a new Tcl command whose name is the same as the path name of the ListNoteBook widget's window. This command may be used to invoke various operations on the widget. It has the following general form:

pathName option ?*arg arg ...*?

PathName is the name of the command, which is the same as the ListNoteBook widget's path name. *Option* and the *arg*s determine the exact behavior of the command. The following commands are possible for

ListNoteBook widgets:

pathName **add** *pageName* ?*option value* ...?

Adds a new ListNotebook page subwidget into the ListNoteBook widget. *pageName* must be the name of an existing entry of the **hlist** subwidget. You must create the entry before calling the **add** command. Please refer to the **tixHList(n)** manual entry for adding entries in an HList widget.

Additional parameters may be supplied to configure this page subwidget. Possible options are:

−createcmd

Specifies a TCL command to be called the first time a page is shown on the screen. This option can be used to delay the creation of the contents of a page until necessary. Therefore, it can be used to speed up interface creation process especially when there are a large number of pages in a ListNoteBook widget.

−raisecmd

Specifies a TCL command to be called whenever this page is raised by the user.

When successful, this command returns the pathname of the newly created page.

pathName **cget** *option*

Returns the current value of the configuration option given by *option*.*Option* may have any of the values accepted by the **tixListNoteBook** command.

pathName **configure** ?*option*? *?value option value ...*?

Query or modify the configuration options of the widget. If no *option* is specified, returns a list describing all of the available options for *pathName* (see **Tk_ConfigureInfo** for information on the format of this list). If *option* is specified with no *value*, then the command returns a list describing the one named option (this list will be identical to the corresponding sublist of the value returned if no *option* is specified). If one or more *option−value* pairs are specified, then the command modifies the given widget option(s) to have the given value(s); in this case the command returns an empty string. *Option* may have any of the values accepted by the **tixListNoteBook** command.

pathName **delete** *pageName*?

Deletes the page identified by *pageName*.

pathName **pagecget** *pageName option*

Returns the current value of the configuration option given by *option* in the page given by *pageName*. *Option* may have any of the values accepted by the **add** widget command.

pathName **pageconfigure** *pageName ?option*? *?value ...*?

When no option is given, prints out the values of all options of this page. If *option* is specified with no *value*, then the command returns the current value of that option. If one or more *option−value* pairs are specified, then the command modifies the given page's option(s) to have the given value(s); in this case the command returns an empty string. *Option* may be any of options accepted by the **add** widget command.

pathName **pages**

Returns a list of the names of all the pages.

pathName **raise** *pageName*

Raise the page identified by *pageName*.

pathName **raised**

Returns the name of the currently raised page.

pathName **subwidget** *name ?args?*

When no options are given, this command returns the pathname of the subwidget of the specified name.

When options are given, the widget command of the specified subwidget will be called with these options.

EXAMPLE

```
tixListNoteBook .n; pack .n
.n subwidget hlist add page1 -text "Page 1"
.n subwidget hlist add page2 -text "Page 2"

set page1 [.n add page1]
set page2 [.n add page2]

button $page1.b -text "On page1"
button $page2.b -text "On page2"

pack $page1.b
pack $page2.b

.n raise page2
```

BINDINGS

When the user activates an entry in the **hlist** subwidget, the page associated with that entry will be raised to the front. This can be done by using the mouse or keyboard. The *hlist* subwidget operates with its **–selectmode** option set to single. See the event bindings of the HList widget for more details.

KEYWORDS

Tix(n), tixHList(n)

NAME

tixMwm - Communicate with the Motif(tm) window manager.

SYNOPSIS

tixMwm *option pathName ?args?*

COMMAND OPTIONS

tixMwm decoration *pathName* ?option? ?value? ?...?

When no options are given, this command returns the values of all the decorations options for the toplevel window with the *pathName*.

When only one option is given without specifying the value, the current value of that option is returned.

When more than one "option value" pairs are passed to this command, the specified values will be assigned to the corresponding options. As a result, the appearance of the Motif decorations around the toplevel window will be changed.

Possible options are: **-border**, **-menu**, **-maximize**, **-minimize**, **-resizeh** and **-title**. The value must be a Boolean value. The values returned by this command are undefined when the window is not managed by mwm.

tixMwm ismwmrunning *pathName*

This returns true if mwm is running on the screen where the specified window is located, false otherwise.

tixMwm protocol *pathName*

When no additional options are given, this command returns all protocols associated with this toplevel window.

tixMwm protocol *pathName* **activate** *protocol_name*

Activate the mwm protocol message in mwm's menu.

tixMwm protocol *pathName* **add** *protocol_name menu_message*

Add a new mwm protocol message for this toplevel window. The message is identified by the string name specified in *protocol_name*. A menu item will be added into mwm's menu as specified by *menu_message*. Once a new mwm protocol message is added to a toplevel, it can be catched by the TK **wm protocol** command. Here is an example:

```
tixMwm protocol . add MY_PRINT_HELLO \
    {"Print Hello"  _H Ctrl<Key>H}
wm protocol . MY_PRINT_HELLO {puts Hello}
```

tixMwm protocol *pathName* **deactivate** *protocol_name*

Deactivate the mwm protocol message in mwm's menu.

tixMwm protocol *pathName* **delete** *protocol_name*

Delete the mwm protocol message from mwm's menu. Please note that the window manager protocol handler associated with this protocol (by the **wm protocol** command) is not deleted automatically. You have to delete the protocol handle explicitly. E.g.:

```
tixMwm protocol . delete MY_PRINT_HELLO
wm protocol . MY_PRINT_HELLO { }
```

BUGS

On some versions of Mwm, the **-border** will not disappear unless **-resizeh** is turned off. Also, the -title will not disappear unless all of **-title**, **-menu**, **-maximize** and **-minimize** are turned off.

KEYWORDS

Tix(n)

NAME

tixNBFrame − Create and manipulate Tix NoteBook Frame widgets

SYNOPSIS

tixNBFrame *pathName ?options*?

SUPER-CLASS

None.

STANDARD OPTIONS

background	**borderWidth**	**cursor**	**disabledForeground**
foreground	**font**		**heighthighlightColor**
highlightThickness	**relief**		**takeFocus**
width			

See the **options(n)** manual entry for details on the standard options.

WIDGET-SPECIFIC OPTIONS

Name: **backPageColor**
Class: **BackPageColor**
Switch: **−backpagecolor**

Specifies the color for the extra space on the row of tabs which is not covered by any page tabs.

Name: **focusColor**
Class: **FocusColor**
Switch: **−focuscolor**

Specifies the color for the focus highlight.

Name: **inactiveBackground**
Class: **InactiveBackground**
Switch: **−inactivebackground**

Specifies the color for the inactive tabs (the active tab always have the same background color as the notebook).

Name: **tabPadX**
Class: **Pad**
Switch: **−tabpadx**

The horizontal padding around the text labels on the page tabs.

Name: **tabPadY**
Class: **Pad**
Switch: **−tabpady**

The vertical padding around the text labels on the page tabs.

DESCRIPTION

The NBFrame widget is used privately inside the **TixNoteBook(n)** widget to display the page tabs. The application programmer should never create a NBFrame widget directly. The sole purpose of this maual page is to describe the options that can be used to configure the appearance of the TixNoteBook widget.

The name of the NBFrame subwidget inside the TixNoteBook widget is called **nbframe**. It can be accessed using the **subwidget** command of the TixNoteBook widget or the **−options** switch:

EXAMPLE

```
tixNoteBook .d -options {
   nbframe.BackPageColor gray60
}
```

```
.d subwidget nbframe config -font fixed

.d add page1 -label "Page1"
set page [.d subwidget page1]
button $page.b1
pack $page.b1

pack .d -expand yes -fill both
```

KEYWORDS

Tix(n), TixNoteBook(n)

NAME

tixNoteBook - Create and manipulate tixNoteBook widgets

SYNOPSIS

tixNoteBook *pathName ?options*?

STANDARD OPTIONS

The NoteBook widget supports all the standard options of a frame widget. See the options(n) manual entry for details on the standard options.

WIDGET-SPECIFIC OPTIONS

Name: **dynamicGeometry**
Class: **DynamicGeometry**
Switch: **–dynamicgeometry**

If set to false, the size of the Notebook will match the size of the largest page. If set to true, the size of the Notebook will match the size of the current page (therefore, the size may change when the user selects different pages). The default value is false. A setting of true is discouraged.

Name: **ipadX**
Class: **Pad**
Switch: **–ipadx**

The amount of internal horizontal paddings around the sides of the page subwidgets.

Name: **ipadY**
Class: **Pad**
Switch: **–ipady**

The amount of internal vertical paddings around the sides of the page subwidgets.

SUBWIDGETS

Name: **nbframe**
Class: **tixNoteBookFrame**

The "note book frame" widget that displays ths tabs of the notebook. Most of the display options of the page tabs are controlled by this subwidget. For example, if you need to choose a different font to display the tab names of the pages, the color of the inactive tabs or the color behind the tabs, you can configure the options of the **nbframe** subwidget. See the manual page of **tixNoteBookFrame(n)** for more details.

In addition, all the page subwidgets created as a result of the **add** command can be accessed by the **subwidget** command. They are identified by the **pageName** parameter to the **add** command.

DESCRIPTION

The **tixNoteBook** command creates a new window (given by the *pathName* argument) and makes it into a NoteBook widget. Additional options, described above, may be specified on the command line or in the option database to configure aspects of the NoteBook widget such as its cursor and relief.

The NoteBook widget can be used to display many windows in a limited space using a "notebook" metaphore. The notebook is divided into a stack of pages (windows). At one time only one of these pages can be shown. The user can navigate through these pages by choosing the visual "tabs" at the top of the NoteBook widget.

WIDGET COMMANDS

The **tixNoteBook** command creates a new Tcl command whose name is the same as the path name of the NoteBook widget's window. This command may be used to invoke various operations on the widget. It has the following general form:

pathName option ?*arg arg ...*?

PathName is the name of the command, which is the same as the NoteBook widget's path name. *Option* and the *arg*s determine the exact behavior of the command. The following commands are possible for NoteBook widgets:

pathName **add** *pageName* ?*option value ...*?

Adds a new notebook page subwidget into the NoteBook widget. Additional parameters may be supplied to configure this page subwidget. Possible options are:

–anchor

Specifies how the information in a tab (e.g. text or a bitmap) is to be displayed in the widget. Must be one of the values **n**, **ne**, **e**, **se**, **s**, **sw**, **w**, **nw**, or **center**. For example, nw means display the information such that its top-left corner is at the top-left corner of the widget.

–bitmap

Specifies a bitmap to display on the tab of this page. The bitmap is displayed only if none of the **-label** or **-image** options are specified.

–createcmd

Specifies a TCL command to be called the first time a page is shown on the screen. This option can be used to delay the creation of the contents of a page until necessary. Therefore, it can be used to speed up interface creation process especially when there are a large number of pages in a NoteBook widget.

–image

Specifies an image to display on the tab of this page. The image is displayed only if the **-label** options is not specified.

–justify

When there are multiple lines of text displayed in a tab, this option determines how the lines line up with each other. Must be one of left, center, or right. **Left** means that the lines' left edges all line up, **center** means that the lines' centers are aligned, and **right** means that the lines' right edges line up.

–label Specifies a text label string to display on the tab of this page subwidget.

–raisecmd

Specifies a TCL command to be called whenever this page is raised by the user.

–state Specifies whether this page can be raised by the user. Must be either **normal** or **disabled**.

–underline

Specifies the integer index of a character to underline in the tab. This option is used by the default bindings to implement keyboard traversal for menu buttons and menu entries. 0 corresponds to the first character of the text displayed in the widget, 1 to the next character, and so on.

–wraplength

This option specifies the maximum line length of the label string on this tab. If the line length of the label string exceeds this length, it is wrapped onto the next line, so that no line is longer than the specified length. The value may be specified in any of the standard forms for screen distances. If this value is less than or equal to 0 then no wrapping is done: lines will break only at newline characters in the text.

pathName **cget** *option*

Returns the current value of the configuration option given by *option.Option* may have any of the values accepted by the **tixNoteBook** command.

pathName **configure** ?*option*? *?value option value ...*?

Query or modify the configuration options of the widget. If no *option* is specified, returns a list describing all of the available options for *pathName* (see **Tk_ConfigureInfo** for information on the format of this list). If *option* is specified with no *value*, then the command returns a list describing the one named option (this list will be identical to the corresponding sublist of the value returned if no *option* is specified). If one or more *option−value* pairs are specified, then the command modifies the given widget option(s) to have the given value(s); in this case the command returns an empty string. *Option* may have any of the values accepted by the **tixNoteBook** command.

pathName **delete** *pageName*?

Deletes the page identified by *pageName*.

pathName **pagecget** *pageName option*

Returns the current value of the configuration option given by *option* in the page given by *pageName*. *Option* may have any of the values accepted by the **add** widget command.

pathName **pageconfigure** *pageName ?option*? *?value ...*?

When no option is given, prints out the values of all options of this page. If *option* is specified with no *value*, then the command returns the current value of that option. If one or more *option−value* pairs are specified, then the command modifies the given page's option(s) to have the given value(s); in this case the command returns an empty string. *Option* may be any of options accepted by the **add** widget command.

pathName **pages**

Returns a list of the names of all the pages.

pathName **raise** *pageName*

Raise the page identified by *pageName*.

pathName **raised**

Returns the name of the currently raised page.

pathName **subwidget** *name ?args?*

When no options are given, this command returns the pathname of the subwidget of the specified name.

When options are given, the widget command of the specified subwidget will be called with these options.

BINDINGS

[1] When the user pressed the left mouse button over a notebook tab, the notebook page associated with that tab will be raised to the top of the stack of pages.

[2] The pages can also be selected using the keyboard. The user can type the **<Tab>** key to cycle among the set of pages. When the focus appears on the desired page, the user can type **<Return>** or **<space>** to select that page. Or, if the user wants to cancel the selection, he/she can type the **<Escape>** key.

KEYWORDS

Tix(n)

NAME

tixOptionMenu – Create and manipulate tixOptionMenu widgets

SYNOPSIS

tixOptionMenu *pathName ?options*?

SUPER-CLASS

The **TixOptionMenu** class is derived from the **TixLabelWidget** class and inherits all the commands, options and subwidgets of its super-class.

STANDARD OPTIONS

The OptionMenu widget supports all the standard Tix widget options. See the **Tix-Options(n)** manual entry for details on the standard Tix widget options.

WIDGET-SPECIFIC OPTIONS

Name: **command**
Class: **Command**
Switch: **–command**

Specifies the command to be called when the **–value** option of the OptionMenu is changed. The command will be called with one arguments -- the new value of the OptionMenu widget.

Name: **disableCallback**
Class: **DisableCallback**
Switch: **–disablecallback**

A boolean value indicating whether callbacks should be disabled. When set to true, the TCL command specified by the **–command** option is not executed when the **–value** of the OptionMenu widget changes.

Name: **dynamicGeometry**
Class: **DynamicGeometry**
Switch: **–dynamicgeometry**

A boolean value indicating whether the size of the **menubutton** subwidget should change dynamically to match the width of the currently selected menu entry. If set to false (the default), the the size of the menubutton subwidget will be wide enough to display every menu entry fully and does not change when the user selects different entries.

Name: **label**
Class: **Label**
Switch: **–label**

Specifies the string to display as the label of this OptionMenu widget.

Name: **labelSide**
Class: **LabelSide**
Switch: **–labelside**

Specifies where the label should be displayed relative to the entry subwidget. Valid options are: **top**, **left**, **right**, **bottom**, **none** or **acrosstop**.

Name: **state**
Class: **State**
Switch: **–state**

Specifies the whether the OptionMenu widget is normal or disabled. Only the values "normal" and "disabled" are recognized.

Name: **value**
Class: **Value**
Switch: **–value**

Specifies the value of the OptionMenu. The value of the OptionMenu widget is the name of the item currently displayed by its **menubutton** subwidget.

Name: **variable**
Class: **Variable**
Switch: **−variable**

Specifies the global variable in which the value of the OptionMenu should be stored. The value of the OptionMenu will be automatically updated when this variable is changed.

SUBWIDGETS

Name: **menu**
Class: **Menu**

The menu subwidget, which is popped up when the user press the **menubutton** subwidget.

Name: **menubutton**
Class: **Menubutton**

The menubutton subwidget.

DESCRIPTION

The **tixOptionMenu** command creates a new window (given by the *pathName* argument) and makes it into a OptionMenu widget. Additional options, described above, may be specified on the command line or in the option database to configure aspects of the OptionMenu such as its cursor and relief.

WIDGET COMMANDS

The **tixOptionMenu** command creates a new Tcl command whose name is the same as the path name of the OptionMenu's window. This command may be used to invoke various operations on the widget. It has the following general form:

pathName option ?*arg arg ...*?

PathName is the name of the command, which is the same as the OptionMenu widget's path name. *Option* and the *arg*s determine the exact behavior of the command. The following commands are possible for OptionMenu widgets:

pathName **add** *type name* ?*option value ...*?

Adds a new item into the OptionMenu widget. *type* must be either **command** or **separator**. The *options* may be any of the valid options for the **command** or **separator** menu entry types for the TK **menu** widget class, except **−command**.

pathName **cget** *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the **tixOptionMenu** command.

pathName **configure** ?*option*? *?value option value ...*?

Query or modify the configuration options of the widget. If no *option* is specified, returns a list describing all of the available options for *pathName* (see **Tk_ConfigureInfo** for information on the format of this list). If *option* is specified with no *value*, then the command returns a list describing the one named option (this list will be identical to the corresponding sublist of the value returned if no *option* is specified). If one or more *option−value* pairs are specified, then the command modifies the given widget option(s) to have the given value(s); in this case the command returns an empty string. *Option* may have any of the values accepted by the **tixOptionMenu** command.

pathName **delete** *name*
Deletes the menu entry identified by *name*.

pathName **disable** *name*
Disables the menu entry identified by *name*.

pathName **enable** *name*
Enables the menu entry identified by *name*.

pathName **entrycget** *name option*
Returns the current value of the configuration option given by *option* in the menu entry identified by *name*. *Option* may have any of the values accepted by the **add** widget command.

pathName **entryconfigure** *name* ?*option*? *?value option value ...*?
Query or modify the configuration options of the menu entry identified by *name*. If no *option* is specified, returns a list describing all of the available options for the menu entry (see **Tk_ConfigureInfo** for information on the format of this list). If *option* is specified with no *value*, then the command returns a list describing the one named option (this list will be identical to the corresponding sublist of the value returned if no *option* is specified). If one or more *option−value* pairs are specified, then the command modifies the given option(s) to have the given value(s); in this case the command returns an empty string. *Option* may have any of the values accepted by the **add** widget command.

pathName **entries**
Returns the names of all the entries currently in the OptionMenu widget.

pathName **subwidget** *name ?args?*
When no options are given, this command returns the pathname of the subwidget of the specified name.

When options are given, the widget command of the specified subwidget will be called with these options.

KEYWORDS
Tix(n)

NAME

tixPanedWindow – Create and manipulate tixPanedWindow widgets

SYNOPSIS

tixPanedWindow *pathName ?options*?

STANDARD OPTIONS

The PanedWindow widget supports all the standard options of a frame widget. See the **options(n)** manual entry for details on the standard options.

WIDGET-SPECIFIC OPTIONS

Name: **command**
Class: **Command**
Switch: **–command**

Specifies the command to invoke when the panes change their sizes. This command is called with a list of integers that record the new sizes of the panes. The sizes of the panes are listed in the order of the panes' creation.

Name: **handleActiveBg**
Class: **HandleActiveBg**
Switch: **–handleactivebg**

Specifies the active background color of the resize handles. When the mouse cursor enters a resize handle, the resize handle will adopt the active background color.

Name: **handleBg**
Class: **Background**
Switch: **–handlebg**

Specifies the normal background color of the resize handles.

Name: **height**
Class: **Height**
Switch: **–height**

Specifies the desired height for the window.

Name: **orientation**
Class: **Orientation**
Switch: **–orientation**
Alias: **–orient**

Specifies the orientation of the panes. Must be either **vertical** or **horizontal**.

Name: **paneBorderWidth**
Class: **PaneBorderWidth**
Switch: **–paneborderwidth**
Alias: **–panebd**

Specifies the border width of the panes.

Name: **paneRelief**
Class: **PaneRelief**
Switch: **–panerelief**

Specifies the border relief of the panes.

Name: **separatorActiveBg**
Class: **SeparatorActiveBg**
Switch: **–separatoractivebg**

Specifies the active background color of the separators. When the user grabs a resize handle, the separators will adopt the active background color.

Name: **separatorBg**
Class: **Background**
Switch: **–separatorbg**

Specifies the normal background color of the separators.

Name: **width**
Class: **Width**
Switch: **–width**

Specifies the desired width for the window.

SUBWIDGETS

All the pane subwidgets created as a result of the **add** command can be accessed by the **subwidget** command. They are identified by the **paneName** parameter to the **add** command.

DESCRIPTION

The **tixPanedWindow** command creates a new window (given by the *pathName* argument) and makes it into a PanedWindow widget. Additional options, described above, may be specified on the command line or in the option database to configure aspects of the PanedWindow widget such as its cursor and relief.

The PanedWindow widget allows the user to interactively manipulate the sizes of several panes. The panes can be arranged either vertically or horizontally. Each individual pane may have upper and lower limits of its size. The user changes the sizes of the panes by dragging the resize handle between two panes.

WIDGET COMMAND

The **tixPanedWindow** command creates a new Tcl command whose name is the same as the path name of the PanedWindow widget's window. This command may be used to invoke various operations on the widget. It has the following general form:

pathName option ?*arg arg ...*?

PathName is the name of the command, which is the same as the frame widget's path name. *Option* and the *arg*s determine the exact behavior of the command. The following commands are possible for PanedWindow widgets:

pathName **add** *paneName* ?*option value ...*?

Adds a new pane subwidget with the name *paneName* into the PanedWindow widget. Additional configuration options can be given to configure the new button subwidget. Three configuration options are supported:

–after *pane*

Specifies that the new pane should be placed after *pane* in the list of panes in this PanedWindow widget. **–at** *integer* Specifies the position of the new pane in the list of panes in this PanedWindow widget. **0** means the first position, **1** means the second, and so on. In addition, **end** means the end of the list.

–before *pane*

Specifies that the new pane should be placed before *pane* in the list of panes in this PanedWindow widget.

–min *integer*

Specifies the minimum size, in pixels, of the new pane; the default is 0.

–max *integer*

Specifies the maximum size, in pixels, of the new pane; the default is 10000.

–size *integer*

Specifies the size, in pixels, of the new pane; if the **–size** option is not given, or set to the empty string, the PanedWindow widget will use the natural size of the pane subwidget.

pathName **cget** *option*

Returns the current value of the configuration option given by *option*. *Option* may be **–min**, **–max** and/or **–size**, or any option accepted by the Tk frame widget.

pathName **configure** ?*option*? ?*value option value ...*?

Query or modify the configuration options of the widget. If no *option* is specified, returns a list describing all of the available options for *pathName* (see **Tk_ConfigureInfo** for information on the format of this list). If *option* is specified with no *value*, then the command returns a list describing the one named option (this list will be identical to the corresponding sublist of the value returned if no *option* is specified). If one or more *option−value* pairs are specified, then the command modifies the given widget option(s) to have the given value(s); in this case the command returns an empty string. *Option* may be any of the non-static options of the PanedWindow widget.

pathName **delete** *paneName*

Removes the pane given by *paneName* and deletes its contents.

pathName **forget** *paneName*

Removes the pane given by *paneName* but does not delete its contents. This pane can be later added back to the PanedWindow widget by the **manage** method.

pathName **manage** *paneName* ?*option value ...*?

Adds the pane given by *paneName* back to the PanedWindow widget. *PaneName* must be already forgotten by the **forget** method. Additional *option-value* pairs, same as those accepted by the **add** method, can be given to control the appearance and position of the pane.

pathName **panecget** *paneName option*

Returns the current value of the configuration option given by *option* in the pane given by *paneName*. *Option* may have any of the values accepted by the **add** widget command.

pathName **paneconfigure** *paneName ?option*? ?*value ...*?

When no option is given, prints out the values of all options of this pane. If *option* is specified with no *value*, then the command returns the current value of that option. If one or more *option−value* pairs are specified, then the command modifies the given pane's option(s) to have the given value(s); in this case the command returns an empty string. *Option* may be **–min**, **–max** and/or **–size**, or any option accepted by the Tk frame widget. The sizes of the panes may be changed as a result of calling the **paneconfigure** command.

pathName **panes**

Returns a list of the names of all panes.

pathName **subwidget** *name ?args?*

When no options are given, returns the pathname of the subwidget of the specified name.

When options are given, the widget command of the specified subwidget will be called with these options.

BINDINGS

The panes' sizes will be changed when the user drags the handles. The change in the panes' sizes may be subjected to the **–min**, **–max** and **–size** options of the panes.

KEYWORDS

TIX, Container Widget

NAME

tixPopupMenu − Create and manipulate tixPopupMenu widgets

SYNOPSIS

tixPopupMenu *pathName ?options*?

SUPER-CLASS

The **tixPopupMenu** class is derived from the **TixShell** class and inherits all the commands, options and subwidgets of its super-class.

STANDARD OPTIONS

The PopupMenu widget supports all the standard options of a frame widget. See the **options(n)** manual entry for details on the standard options.

WIDGET-SPECIFIC OPTIONS

Name: **state**
Class: **State**
Switch: **−state**

Must be either **disabled** or **normal**. The PopupMenu widget will not pop up unless its **−state** is set to **normal**.

Name: **title**
Class: **Title**
Switch: **−title**

Specifies a text string to display inside the **menubutton** subwidget, as the title of this PopupMenu.

SUBWIDGETS

Name: **menu**
Class: **Menu**

The menu subwidget.

Name: **menubutton**
Class: **Menubutton**

The menubutton subwidget.

DESCRIPTION

The **tixPopupMenu** command creates a new window (given by the *pathName* argument) and makes it into a PopupMenu widget. Additional options, described above, may be specified on the command line or in the option database to configure aspects of the PopupMenu widget such as its cursor and relief.

The Tix PopupMenu widget can be used as a replacement of the **tk_popup** command. The advantage of the Tix PopupMenu widget is it requires less application code to manipulate. Also, it provides a title for the popup menu, which is not available from **tk_popup**.

WIDGET COMMANDS

The **tixPopupMenu** command creates a new Tcl command whose name is the same as the path name of the PopupMenu widget's window. This command may be used to invoke various operations on the widget. It has the following general form:

pathName option ?*arg arg ...*?

PathName is the name of the command, which is the same as the PopupMenu widget's path name. *Option* and the *arg*s determine the exact behavior of the command. The following commands are possible for PopupMenu widgets:

pathName **bind** *widget* ?*widget ...*?

Binds this PopupMenu to one or more *widgets*. The PopupMenu will be activated when the user presses the right mouse button over these *widgets*.

pathName **cget** *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the **tixPopupMenu** command.

pathName **configure** ?*option*? *?value option value ...*?

Query or modify the configuration options of the widget. If no *option* is specified, returns a list describing all of the available options for *pathName* (see **Tk_ConfigureInfo** for information on the format of this list). If *option* is specified with no *value*, then the command returns a list describing the one named option (this list will be identical to the corresponding sublist of the value returned if no *option* is specified). If one or more *option−value* pairs are specified, then the command modifies the given widget option(s) to have the given value(s); in this case the command returns an empty string. *Option* may have any of the values accepted by the **tixPopupMenu** command.

pathName **post** *widget x y*

Posts the PopupMenu inside the *widget* at the coordinate *x*,*y*.

pathName **unbind** *widget* ?*widget ...*?

Cancels the PopupMenu's binding with the *widget(s)*.

pathName **subwidget** *name ?args?*

When no options are given, this command returns the pathname of the subwidget of the specified name.

When options are given, the widget command of the specified subwidget will be called with these options.

KEYWORDS

Tix(n)

NAME

tixScrolledHList − Create and manipulate Tix ScrolledHList widgets

SYNOPSIS

tixScrolledHList *pathName ?options*?

STANDARD OPTIONS

anchor	**background**	**cursor**
relief	**borderWidth**	

See the **options(n)** manual entry for details on the standard options.

WIDGET-SPECIFIC OPTIONS

Name: **height**
Class: **Height**
Switch: **-height**

Specifies the desired height for the window, in pixels.

Name: **scrollbar**
Class: **Scrollbar**
Switch: **−scrollbar**

Specifies the display policy of the scrollbars. The following values are recognized:

auto ?+*x*? ?−*x*? ?+*y*? ?−*y*?
When **−scrollbar** is set to "**auto**", the scrollbars are shown only when needed. Additional modifiers can be used to force a scrollbar to be shown or hidden. For example, "**auto −y**" means the horizontal scrollbar should be shown when needed but the vertical scrollbar should always be hidden; "**auto +x**" means the vertical scrollbar should be shown when needed but the horizontal scrollbar should always be shown, and so on.

both Both scrollbars are shown

none The scrollbars are never shown.

x Only the horizontal scrollbar is shown;

y Only the vertical scrollbar is shown.

Name: **width**
Class: **Width**
Switch: **-width**

Specifies the desired width for the window, in pixels.

SUBWIDGETS

Name: **hsb**
Class: **Scrollbar**

The horizontal scrollbar subwidget.

Name: **hlist**
Class: **Hlist**

The tixHList subwidget inside the ScrolledHList widget.

Name: **vsb**
Class: **Scrollbar**

The vertical scrollbar subwidget.

DESCRIPTION

The **tixScrolledHList** command creates a new window (given by the *pathName* argument) and makes it into a ScrolledHList widget. Additional options, described above, may be specified on the command line or in the option database to configure aspects of the ScrolledHList widget such as its cursor and relief.

WIDGET COMMANDS

The **tixScrolledHList** command creates a new Tcl command whose name is the same as the path name of the ScrolledHList widget's window. This command may be used to invoke various operations on the widget. It has the following general form:

pathName option ?*arg arg ...*?

PathName is the name of the command, which is the same as the ScrolledHList widget's path name. *Option* and the *arg*s determine the exact behavior of the command. The following commands are possible for ScrolledHList widgets:

pathName **cget** *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the **tixScrolledHList** command.

pathName **configure** ?*option*? *?value option value ...*?

Query or modify the configuration options of the widget. If no *option* is specified, returns a list describing all of the available options for *pathName* (see **Tk_ConfigureInfo** for information on the format of this list). If *option* is specified with no *value*, then the command returns a list describing the one named option (this list will be identical to the corresponding sublist of the value returned if no *option* is specified). If one or more *option−value* pairs are specified, then the command modifies the given widget option(s) to have the given value(s); in this case the command returns an empty string. *Option* may have any of the values accepted by the **tixScrolledHList** command.

pathName **subwidget** *name ?args?*

When no additional arguments are given, returns the pathname of the subwidget of the specified name.

When no additional arguments are given, the widget command of the specified subwidget will be called with these parameters.

KEYWORDS

Tix(n)

NAME

tixScrolledListBox − Create and manipulate Tix ScrolledListBox widgets

SYNOPSIS

tixScrolledListBox *pathName ?options*?

STANDARD OPTIONS

anchor	**background**	**cursor**
relief	**borderWidth**	

See the **options(n)** manual entry for details on the standard options.

WIDGET-SPECIFIC OPTIONS

Name: **anchor**
Class: **Anchor**
Switch: **−anchor**

Specifies the alignment of the items inside the listbox subwidget. Only the values **w** and **e** are allowed. When set to **w**, the listbox is automatically aligned to the beginning of the items. When set to **e**, the listbox is automatically aligned to the end of the items. Automatically alignment only happens when the ScrolledListBox widget changes its size.

Name: **browsecmd**
Class: **BrowseCmd**
Switch: **−browsecmd**

Specifies the command to be called when the user browses the elements inside the **listbox** subwidget (see the BINDINGS section below).

Name: **command**
Class: **Command**
Switch: **−command**

Specifies the command to be called when the user invokes the **listbox** subwidget (see the BINDINGS section below).

Name: **height**
Class: **Height**
Switch: **−height**

Specifies the desired height for the window, in pixels.

Name: **scrollbar**
Class: **Scrollbar**
Switch: **−scrollbar**

Specifies the display policy of the scrollbars. The following values are recognized:

auto ?+*x*? ?−*x*? ?+*y*? ?−*y*?
When **−scrollbar** is set to "**auto**", the scrollbars are shown only when needed. Additional modifiers can be used to force a scrollbar to be shown or hidden. For example, "**auto −y**" means the horizontal scrollbar should be shown when needed but the vertical scrollbar should always be hidden; "**auto +x**" means the vertical scrollbar should be shown when needed but the horizontal scrollbar should always be shown, and so on.

both Both scrollbars are shown

none The scrollbars are never shown.

x Only the horizontal scrollbar is shown;

y Only the vertical scrollbar is shown.

Name: **width**
Class: **Width**
Switch: **-width**

Specifies the desired width for the window, in pixels.

SUBWIDGETS

Name: **hsb**
Class: **Scrollbar**

The horizontal scrollbar subwidget.

Name: **listbox**
Class: **Listbox**

The listbox subwidget inside the ScrolledListBox widget.

Name: **vsb**
Class: **Scrollbar**

The vertical scrollbar subwidget.

DESCRIPTION

The **tixScrolledListBox** command creates a new window (given by the *pathName* argument) and makes it into a ScrolledListBox widget. Additional options, described above, may be specified on the command line or in the option database to configure aspects of the ScrolledListBox widget such as its cursor and relief.

WIDGET COMMANDS

The **tixScrolledListBox** command creates a new Tcl command whose name is the same as the path name of the ScrolledListBox widget's window. This command may be used to invoke various operations on the widget. It has the following general form:

pathName option ?*arg arg ...*?

PathName is the name of the command, which is the same as the ScrolledListBox widget's path name. *Option* and the *arg*s determine the exact behavior of the command. The following commands are possible for ScrolledListBox widgets:

pathName **cget** *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the **tixScrolledListBox** command.

pathName **configure** ?*option*? *?value option value ...*?

Query or modify the configuration options of the widget. If no *option* is specified, returns a list describing all of the available options for *pathName* (see **Tk_ConfigureInfo** for information on the format of this list). If *option* is specified with no *value*, then the command returns a list describing the one named option (this list will be identical to the corresponding sublist of the value returned if no *option* is specified). If one or more *option−value* pairs are specified, then the command modifies the given widget option(s) to have the given value(s); in this case the command returns an empty string. *Option* may have any of the values accepted by the **tixScrolledListBox** command.

pathName **subwidget** *name ?args?*

When no additional arguments are given, returns the pathname of the subwidget of the specified name.

When no additional arguments are given, the widget command of the specified subwidget will be called with these parameters.

BINDINGS

[1] If the **−browsecmd** option is set, the command which it referes to is called whenever a <ButtonPress−1> or a <Motion−1> event occurrs inside the **listbox** subwidget.

[2] If the **−selectmode** option of the **listbox** subwidget is **browse**, the command specified by the **−command** option is invoked when a <ButtonRelease−1> event occurrs inside the **listbox** subwidget. If the **−selectmode** option of the **listbox** subwidget is **single**, **multiple** or **extended**, the command specified by the **−command** option is invoked when a <Double−1> event occurrs inside the **listbox** subwidget.

BUGS

The capitalization of some of the commands names in Tix 3.x has been changed in Tix 4.0. All commands that ended with **box** have been changed to a capitalized **Box**. Hence, the command **tixScrolledListbox** in Tix 3.x has been changed to **tixScrolledListBox** in Tix 4.0

KEYWORDS

Tix(n)

NAME

tixScrolledText – Create and manipulate Tix ScrolledText widgets

SYNOPSIS

tixScrolledText *pathName ?options*?

STANDARD OPTIONS

anchor	**background**	**cursor**
relief	**borderWidth**	

See the **options(n)** manual entry for details on the standard options.

WIDGET-SPECIFIC OPTIONS

Name: **height**
Class: **Height**
Switch: **-height**

Specifies the desired height for the window, in pixels.

Name: **scrollbar**
Class: **Scrollbar**
Switch: **–scrollbar**

Specifies the display policy of the scrollbars. The following values are recognized:

auto ?+*x*? ?−*x*? ?+*y*? ?−*y*?

When **–scrollbar** is set to "**auto**", the scrollbars are shown only when needed. Additional modifiers can be used to force a scrollbar to be shown or hidden. For example, "**auto –y**" means the horizontal scrollbar should be shown when needed but the vertical scrollbar should always be hidden; "**auto +x**" means the vertical scrollbar should be shown when needed but the horizontal scrollbar should always be shown, and so on.

both Both scrollbars are shown

none The scrollbars are never shown.

x Only the horizontal scrollbar is shown;

y Only the vertical scrollbar is shown.

Name: **width**
Class: **Width**
Switch: **-width**

Specifies the desired width for the window, in pixels.

SUBWIDGETS

Name: **hsb**
Class: **Scrollbar**

The horizontal scrollbar subwidget.

Name: **text**
Class: **Text**

The Text subwidget inside the ScrolledText widget.

Name: **vsb**
Class: **Scrollbar**

The vertical scrollbar subwidget.

DESCRIPTION

The **tixScrolledText** command creates a new window (given by the *pathName* argument) and makes it into a ScrolledText widget. Additional options, described above, may be specified on the command line or in the option database to configure aspects of the ScrolledText widget such as its cursor and relief.

WIDGET COMMANDS

The **tixScrolledText** command creates a new Tcl command whose name is the same as the path name of the ScrolledText widget's window. This command may be used to invoke various operations on the widget. It has the following general form:

pathName option ?*arg arg ...*?

PathName is the name of the command, which is the same as the ScrolledText widget's path name. *Option* and the *arg*s determine the exact behavior of the command. The following commands are possible for ScrolledText widgets:

pathName **cget** *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the **tixScrolledText** command.

pathName **configure** ?*option*? *?value option value ...*?

Query or modify the configuration options of the widget. If no *option* is specified, returns a list describing all of the available options for *pathName* (see **Tk_ConfigureInfo** for information on the format of this list). If *option* is specified with no *value*, then the command returns a list describing the one named option (this list will be identical to the corresponding sublist of the value returned if no *option* is specified). If one or more *option−value* pairs are specified, then the command modifies the given widget option(s) to have the given value(s); in this case the command returns an empty string. *Option* may have any of the values accepted by the **tixScrolledText** command.

pathName **subwidget** *name ?args?*

When no additional arguments are given, returns the pathname of the subwidget of the specified name.

When no additional arguments are given, the widget command of the specified subwidget will be called with these parameters.

KEYWORDS

Tix(n)

NAME

tixScrolledWindow − Create and manipulate Tix ScrolledWindow widgets

SYNOPSIS

tixScrolledWindow *pathName ?options*?

STANDARD OPTIONS

anchor	**background**	**cursor**
relief	**borderWidth**	

See the **options(n)** manual entry for details on the standard options.

WIDGET-SPECIFIC OPTIONS

Name: **height**
Class: **Height**
Switch: **-height**

Specifies the desired height for the window, in pixels.

Name: **scrollbar**
Class: **Scrollbar**
Switch: **−scrollbar**

Specifies the display policy of the scrollbars. The following values are recognized:

auto ?+*x*? ?−*x*? ?+*y*? ?−*y*?
When **−scrollbar** is set to "**auto**", the scrollbars are shown only when needed. Additional modifiers can be used to force a scrollbar to be shown or hidden. For example, "**auto −y**" means the horizontal scrollbar should be shown when needed but the vertical scrollbar should always be hidden; "**auto +x**" means the vertical scrollbar should be shown when needed but the horizontal scrollbar should always be shown, and so on.

both Both scrollbars are shown

none The scrollbars are never shown.

x Only the horizontal scrollbar is shown;

y Only the vertical scrollbar is shown.

Name: **width**
Class: **Width**
Switch: **-width**

Specifies the desired width for the window, in pixels.

Name: **xScrollIncrement**
Class: **ScrollIncrement**
Switch: **-xscrollincrement**

Specifies by how much the window should be scrolled in the horizontal direction when the user presses the arrows in the horizontal scrollbar. In Pixels.

Name: **yScrollIncrement**
Class: **ScrollIncrement**
Switch: **-yscrollincrement**

Specifies by how much the window should be scrolled in the vertical direction when the user presses the arrows in the horizontal scrollbar. In pixels.

SUBWIDGETS

Name: **hsb**
Class: **Scrollbar**

The horizontal scrollbar subwidget.

Name: **window**
Class: **Frame**

The frame subwidget which is scrolled by the ScrolledWindow widget.

Name: **vsb**
Class: **Scrollbar**

The vertical scrollbar subwidget.

DESCRIPTION

The **tixScrolledWindow** command creates a new window (given by the *pathName* argument) and makes it into a ScrolledWindow widget. Additional options, described above, may be specified on the command line or in the option database to configure aspects of the ScrolledWindow widget such as its cursor and relief.

CREATING WIDGETS INSIDE A SCROLLEDWINDOW WIDGET

To create widgets inside a ScrolledWindow widget, one must create the new widgets relative to the **window** subwidget and manage them inside the **window** subwidget. An error will be generated if one tries to create widgets as immediate children of the ScrolledWindow. For example: the following is correct code, which creates new widgets inside the window subwidget:

```
tixScrolledWindow .w; pack .w
set f [.w subwidget window]
button $f.b −text hi −width 40 −height 40
pack $f.b
```

The following example code is *incorrect* because it tries to create immediate children of the ScrolledWindow **.w**:

```
tixScrolledWindow .w; pack .w
button .w.b −text hi −width 40 −height 40
pack .w.b
```

WIDGET COMMANDS

The **tixScrolledWindow** command creates a new Tcl command whose name is the same as the path name of the ScrolledWindow widget's window. This command may be used to invoke various operations on the widget. It has the following general form:

pathName option ?*arg arg ...*?

PathName is the name of the command, which is the same as the ScrolledWindow widget's path name. *Option* and the *arg*s determine the exact behavior of the command. The following commands are possible for ScrolledWindow widgets:

pathName **cget** *option*
: Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the **tixScrolledWindow** command.

pathName **configure** ?*option*? ?*value option value ...*?
: Query or modify the configuration options of the widget. If no *option* is specified, returns a list describing all of the available options for *pathName* (see **Tk_ConfigureInfo** for information on the format of this list). If *option* is specified with no *value*, then the command returns a list describing the one named option (this list will be identical to the corresponding sublist of the value returned if no *option* is specified). If one or more *option−value* pairs are specified, then the command modifies the given widget option(s) to have the given value(s); in this case the command

returns an empty string. *Option* may have any of the values accepted by the **tixScrolledWindow** command.

pathName **subwidget** *name ?args?*

When no additional arguments are given, returns the pathname of the subwidget of the specified name.

When no additional arguments are given, the widget command of the specified subwidget will be called with these parameters.

KEYWORDS

Tix(n)

NAME

tixSelect − Create and manipulate tixSelect widgets

SYNOPSIS

tixSelect *pathName ?options*?

SUPER-CLASS

The **TixSelect** class is derived from the **TixLabelWidget** class and inherits all the commands, options and subwidgets of its super-class.

STANDARD OPTIONS

The Select widget supports all the standard options of a frame widget. See the **options(n)** manual entry for details on the standard options.

WIDGET-SPECIFIC OPTIONS

Name: **allowZero**
Class: **AllowZero**
Switch: **−allowzero**

A boolean value that specifies whether the selection can be empty. When set to false, at least one button subwidget must be selected at any time.

Note: When the Select widget is first constructed, the default selection is always empty, even if **−allowzero** is set to **false**.

Name: **buttonType**
Class: **ButtonType**
Switch: **−buttontype**

The type of buttons to be used as subwidgets inside the Select widget. By default, the standard Tk **button** widget class is used.

Name: **command**
Class: **Command**
Switch: **−command**

Specifies the TCL command to be executed when the **−value** of the Select widget is changed. This command will be invoked with two arguments. The first is the name of the button subwidget that has toggled. The second is a boolean value indicating whether the button subwidget is selected. This command is executed only when the **−disableCallback** option is set to false.

Name: **disableCallback**
Class: **DisableCallback**
Switch: **−disablecallback**

A boolean value indicating whether callbacks should be disabled. When set to true, the TCL command specified by the **−command** option is not executed when the **−value** of the Select widget changes.

Name: **orientation**
Class: **Orientation**
Switch: **−orientation**
Alias: **−orient**

Specifies the orientation of the button subwidgets. Only the values **horizontal** and **vertical** are recognized. This is a *static option* and it can only be assigned during the creation of the widget.

Name: **label**
Class: **Label**
Switch: **−label**

Specifies the string to display as the label of this Select widget.

Name: **labelSide**
Class: **LabelSide**
Switch: **–labelside**

Specifies where the label should be displayed relative to the Select widget. Valid options are: **top**, **left**, **right**, **bottom**, **none** or **acrosstop**.

Name: **padX**
Class: **Pad**
Switch: **–padx**

Specifies the horizontal padding between two neighboring button subwidgets. This is a *static option* and it can only be assigned during the creation of the widget.

Name: **padY**
Class: **Pad**
Switch: **–padx**

Specifies the vertical padding between two neighboring button subwidgets. This is a *static option* and it can only be assigned during the creation of the widget.

Name: **radio**
Class: **Radio**
Switch: **–radio**

A boolean value that specifies whether the Select widget should act as a radio-box. When set to true, at most one button subwidget can be selected at any time. This is a *static option* and it can only be assigned during the creation of the widget.

Name: **selectedBg**
Class: **SelectedBg**
Switch: **–selectedbg**

Specifies the background color of all the selected button subwidgets.

Name: **state**
Class: **State**
Switch: **–state**

Specifies the state of all the buttons inside the Select widget. Only the values **normal** and **disabled** are recognized. When the state is set to **disabled**, all user actions on this Select widget are ignore.

Name: **validateCmd**
Class: **ValidateCmd**
Switch: **–validatecmd**

Specifies a TCL command to be called when the -value of the Select widget is about to change. This command is called with one parameter -- the new **–value** entered by the user. This command is to validate this new value by returning a value it deems valid.

Name: **value**
Class: **Value**
Switch: **–value**

The value of a Select widget is a list of the names of the button subwidgets that have been selected by the user.

When you assign the value of a Select widget using the "config -value" widget command, the TCL command specified by the **–command** option will be invoked if some button subwidgets are toggled.

Name: **variable**
Class: **Variable**
Switch: **−variable**

Specifies the global variable in which the value of the Select widget should be stored. The value of a Select widget is stored as a list of the names of the button subwidgets that have been selected by the user. The value of the Select widget will be automatically updated when this variable is changed.

SUBWIDGETS

Name: **label**
Class: **Label**

The label subwidget.

In addition, all the button subwidgets created as a result of the **add** widget command can be accessed by the **subwidget** command. They are identified by the *buttonName* parameter to the **add** widget command. Here is an example:

```
tixSelect .s
pack .s
.s add eat   -text Eat
.s add sleep -text Sleep
.s subwidget eat   config -fg green
.s subwidget sleep config -fg red
```

DESCRIPTION

The **tixSelect** command creates a new window (given by the *pathName* argument) and makes it into a Select widget. Additional options, described above, may be specified on the command line or in the option database to configure aspects of the Select widget such as its cursor and relief.

The Select widget is a container of button subwidgets. It can be used to provide radio-box or check-box style of selection options for the user.

WIDGET COMMANDS

The **tixSelect** command creates a new Tcl command whose name is the same as the path name of the Select widget's window. This command may be used to invoke various operations on the widget. It has the following general form:

pathName option ?*arg arg ...*?

PathName is the name of the command, which is the same as the Select widget's path name. *Option* and the *arg*s determine the exact behavior of the command. The following commands are possible for Select widgets:

pathName **add** *buttonName* ?*option value ...* ?

Adds a new button subwidget with the name buttonName into the Select widget. Additional configuration options can be given to configure the new button subwidget.

pathName **cget** *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the **tixSelect** command.

pathName **configure** ?*option*? ?*value option value ...*?

Query or modify the configuration options of the widget. If no *option* is specified, returns a list describing all of the available options for *pathName* (see **Tk_ConfigureInfo** for information on the format of this list). If *option* is specified with no *value*, then the command returns a list describing the one named option (this list will be identical to the corresponding sublist of the value returned if no *option* is specified). If one or more *option−value* pairs are specified, then the command modifies the given widget option(s) to have the given value(s); in this case the command returns an empty string. *Option* may have any of the values accepted by the **tixSelect** command.

pathName **invoke** *buttonName*

Invokes the button subwidget with the name buttonName.

pathName **subwidget** *name ?args?*

When no options are given, returns the pathname of the subwidget of the specified name.

When options are given, the widget command of the specified subwidget will be called with these options.

BINDINGS

When the user presses the left mouse button over the a button subwidget, it will be toggled and the **−value** option of the tixSelect widget will be changed.

EXAMPLE

The following example creates a radio-box style iconbar for the user to choose one value among **eat**, **work** or **sleep**.

```
tixSelect .s −radio true −allowzero false
.s add eat   −bitmap [tix getbitmap eat]
.s add work  −bitmap [tix getbitmap work]
.s add sleep −bitmap [tix getbitmap sleep]
```

KEYWORDS

Tix(n), Container Widget

NAME

tixStdButonBox − Create and manipulate Tix StdButonBox widgets

SYNOPSIS

tixStdButonBox *pathName ?options*?

STANDARD OPTIONS

anchor	**background**	**cursor**
relief	**borderWidth**	

See the **options(n)** manual entry for details on the standard options.

WIDGET-SPECIFIC OPTIONS

Name: **orientation**
Class: **Orientation**
Switch: **−orientation**
Alias: **−orient**

Static Option. Specifies the orientation of the button subwidgets. Only the values "horizontal" and "vertical" are recognized.

Name: **padx**
Class: **Pad**
Switch: **−padx**

Specifies the horizontal padding between two neighboring button subwidgets in the StdButonBox widget.

Name: **padx**
Class: **Pad**
Switch: **−padx**

Specifies the vertical padding between two neighboring button subwidgets in the StdButonBox widget.

Name: **state**
Class: **State**
Switch: **−state**

Specifies the state of all the buttons inside the StdButtonBox widget.

Note: Setting this option using the *config* widget command will enable or disable all the buttons subwidgets. Original states of the individual buttons are *not* saved.

SUBWIDGETS

Name: **ok**
Class: **Button**

The first button subwidget. By default it displays the text string "Ok"

Name: **apply**
Class: **Button**

The second button subwidget. By default it displays the text string "Apply"

Name: **cancel**
Class: **Button**

The third button subwidget. By default it displays the text string "Cancel"

Name: **help**

Class: **Button**

The fourth button subwidget. By default it displays the text string "Help"

DESCRIPTION

The **tixStdButonBox** command creates a new window (given by the *pathName* argument) and makes it into a StdButonBox widget. Additional options, described above, may be specified on the command line or in the option database to configure aspects of the StdButonBox such as its cursor and relief.

The StdButonBox widget is a group of Standard buttons for Motif-like dialog boxes.

WIDGET COMMAND

The **tixStdButonBox** command creates a new Tcl command whose name is the same as the path name of the StdButonBox's window. This command may be used to invoke various operations on the widget. It has the following general form:

pathName option ?*arg arg ...*?

PathName is the name of the command, which is the same as the StdButonBox widget's path name. *Option* and the *args* determine the exact behavior of the command. The following commands are possible for StdButonBox widgets:

pathName **cget** *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the **tixStdButonBox** command.

pathName **configure** ?*option*? *?value option value ...*?

Query or modify the configuration options of the widget. If no *option* is specified, returns a list describing all of the available options for *pathName* (see **Tk_ConfigureInfo** for information on the format of this list). If *option* is specified with no *value*, then the command returns a list describing the one named option (this list will be identical to the corresponding sublist of the value returned if no *option* is specified). If one or more *option−value* pairs are specified, then the command modifies the given widget option(s) to have the given value(s); in this case the command returns an empty string. *Option* may have any of the values accepted by the **tixStdButonBox** command.

pathName **invoke** *buttonName*

Invoke the button subwidget with the name buttonName.

pathName **subwidget** *name ?args?*

When no additional arguments are given, returns the pathname of the subwidget of the specified name.

When no additional arguments are given, the widget command of the specified subwidget will be called with these parameters.

BINDINGS

TixStdButonBox widgets have no default bindings. The button subwidgets retain their default Tk bindings.

KEYWORDS

Tix(n), Container Widgets

NAME

TixIntro − Introduction to the Tix widget set

DESCRIPTION

Tix is a set of mega widgets based on the standard Tk widgets. If you are planning only to use Tix with the standard Tk widget set, you can use the program **tixwish(1)** to interpret your TCL scripts.

To use Tix with other TCL extension packages, you have to call the function **Tix_Init()** in your **Tcl_AppInit()** function. Here is an example:

```
int Tcl_AppInit(interp)
   Tcl_Interp *interp;
{
   Tk_Window main;

   main = Tk_MainWindow(interp);

   if (Tcl_Init(interp) == TCL_ERROR) {
          return TCL_ERROR;
   }
   if (Tk_Init(interp) == TCL_ERROR) {
          return TCL_ERROR;
   }

   if (Tix_Init(interp) == TCL_ERROR) {
          return TCL_ERROR;
   }
   /*
    * Call the init procedures for included packages.
    * Each call should look like this:
    *
    * if (Mod_Init(interp) == TCL_ERROR) {
    *     return TCL_ERROR;
    * }
    *
    * where "Mod" is the name of the module.
    */
}
```

Files

The release notes of this version of Tix is in the HTML file **Tix4.0/README.html**. Plain text version of this file can be found as **Tix4.0/README.txt**. Latest information about Tix can also be located on line at <URL:http://www.xpi.com/tix/>

KEYWORDS

Tix(n), compound widgets, Tix Intrinsics

NAME

tixTree − Create and manipulate tixTree widgets

SYNOPSIS

tixTree *pathName ?options*?

SUPER-CLASS

The **TixTree** class is derived from the **TixScrolledHList** class and inherits all the commands, options and subwidgets of its super-class.

STANDARD OPTIONS

TixTree supports all the standard options of a frame widget. See the **options(n)** manual entry for details on the standard options.

WIDGET-SPECIFIC OPTIONS

Name: **browseCmd**
Class: **BrowseCmd**
Switch: **−browsecmd**

Specifies a command to call whenever the user browses on an entry (usually by single-clicking on the entry). The command is called with one argument, the pathname of the entry.

Name: **closeCmd**
Class: **CloseCmd**
Switch: **−closecmd**

Specifies a command to call whenever an entry needs to be closed (See the BINDINGS section below). This command is called with one argument, the pathname of the entry. This command should perform appropriate actions to close the specified entry. If the **−closecmd** option is not specified, the default closing action is to hide all child entries of the specified entry.

Name: **command**
Class: **Command**
Switch: **−command**

Specifies a command to call whenever the user activates an entry (usually by double-clicking on the entry). The command is called with one argument, the pathname of the entry.

Name: **ignoreInvoke**
Class: **IgnoreInvoke**
Switch: **−ignoreinvoke**

A Boolean value that specifies when a branch should be opened or closed. A branch will always be opened or closed when the user presses the (+) and (-) indicators. However, when the user invokes a branch (by doublc-clicking or pressing <Return>), the branch will be opened or closed only if **−ignoreinvoke** is set to false (the default setting).

Name: **openCmd**
Class: **OpenCmd**
Switch: **−opencmd**

Specifies a command to call whenever an entry needs to be opened (See the BINDINGS section below). This command is called with one argument, the pathname of the entry. This command should perform appropriate actions to open the specified entry. If the **−opencmd** option is not specified, the default opening action is to show all the child entries of the specified entry.

SUBWIDGETS

Name: **hlist**
Class: **TixHList**

The hierarchical listbox that displays the tree.

Name: **hsb**
Class: **Scrollbar**

The horizontal scrollbar subwidget.

Name: **vsb**
Class: **Scrollbar**

The vertical scrollbar subwidget.

DESCRIPTION

The **tixTree** command creates a new window (given by the *pathName* argument) and makes it into a Tree widget. Additional options, described above, may be specified on the command line or in the option database to configure aspects of the Tree widget such as its cursor and relief.

The Tree widget can be used to display hierachical data in a tree form. The user can adjust the view of the tree by opening or closing parts of the tree.

To display a static tree structure, you can add the entries into the **hlist** subwidget and hide any entries as desired. Then you can call the **autosetmode** method. This will set up the Tree widget so that it handles all the *open* and *close* events automatically. (Please see the demonstration program demos/samples/Tree.tcl).

The above method is not applicable if you want to maintain a dynamic tree structure, i.e, you do not know all the entries in the tree and you need to add or delete entries subsequently. To do this, you should first create the entries in the **hlist** subwidget. Then, use the setmode method to indicate the entries that can be opened or closed, and use the **−opencmd** and **−closecmd** options to handle the opening and closing events. (Please see the demonstration program demos/samples/DynTree.tcl demo).

WIDGET COMMANDS

The **tixTree** command creates a new Tcl command whose name is the same as the path name of the Tree's window. This command may be used to invoke various operations on the widget. It has the following general form:

pathName option ?*arg arg ...*?

PathName is the name of the command, which is the same as the Tree widget's path name. *Option* and the *arg*s determine the exact behavior of the command. The following commands are possible for Tree widgets:

pathName **autosetmode**
: This command calls the **setmode** method for all the entries in this Tree widget: if an entry has no child entries, its mode is set to **none**. Otherwise, if the entry has any hidden child entries, its mode is set to **open**; otherwise its mode is set to **close**.

pathName **cget** *option*
: Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the **tixTree** command.

pathName **close** *entryPath*
: Close the entry given by *entryPath* if its *mode* is **close**.

pathName **configure** ?*option*? ?*value option value ...*?
: Query or modify the configuration options of the widget. If no *option* is specified, returns a list describing all of the available options for *pathName* (see **Tk_ConfigureInfo** for information on the format of this list). If *option* is specified with no *value*, then the command returns a list describing the one named option (this list will be identical to the corresponding sublist of the value

returned if no *option* is specified). If one or more *option−value* pairs are specified, then the command modifies the given widget option(s) to have the given value(s); in this case the command returns an empty string. *Option* may have any of the values accepted by the **tixTree** command.

pathName **getmode** *entryPath*

Returns the current *mode* of the entry given by *entryPath*.

pathName **open** *entryPath*

Open the entry givaen by *entryPath* if its *mode* is **open**.

pathName **setmode** *entryPath mode*

This command is used to indicate whether the entry given by *entryPath* has children entries and whether the children are visible. *mode* must be one of **open**, **close** or **none**. If *mode* is set to **open**, a (+) indicator is drawn next the the entry. If *mode* is set to **close**, a (-) indicator is drawn next the the entry. If *mode* is set to **none**, no indicators will be drawn for this entry. The default *mode* is none. The **open** mode indicates the entry has hidden children and this entry can be opened by the user. The **close** mode indicates that all the children of the entry are now visible and the entry can be closed by the user.

pathName **subwidget** *name ?args?*

When no options are given, this command returns the pathname of the subwidget of the specified name.

When options are given, the widget command of the specified subwidget will be called with these options.

BINDINGS

The basic mouse and keyboard bindings of the Tree widget is the same as the bindings of the HList widget.

In addition, the entries can be opened or closed under the following conditions:

[1] If the *mode* of the entry is **open**, it can be opened by clicking on its (+) indicator or double-clicking on the entry.

[2] If the *mode* of the entry is **close**, it can be closed by clicking on its (-) indicator or double-clicking on the entry.

KEYWORDS

Tix(n),tixHList(n)

NAME

Utils - Utility commands in Tix 4.0.

SYNOPSIS

tixDescendants *pathName*

tixDisableAll *pathName*

tixEnableAll *pathName*

tixPushGrab ?-*global*? *window*

tixPopGrab

DESCRIPTION

tixDescendants *pathName*
Returns a list of all the descendant widgets of *pathName* plus *pathName* itself.

tixDisableAll *pathName*
Disables *pathName* and all its descendants.

tixEnableAll *pathName*
Enables *pathName* and all its descendants.

tixPushGrab ?-*global*? *window*
The **tixPushGrab** and **tixPopGrab** commands allows you to perform "cascade-grabbing". **tixPushGrab** calls the **grab(n)** command on *window* and saves *window* on a grabbing stack.

tixPopGrab
tixPopGrab pops the top-most element from the grabbing stack and release its grab. If the grabbing stack is not empty, then **tixPopGrab** will execute grab(n) on the current top-most element in the grabbing stack.

NOTES

Some Tix widgets (for example, tixComboBox and tixPanedWindow) grabs the screen on certain occasions using **tixPushGrab** and **tixPopGrab**. Therefore, if you need to grab the screen when these widgets are present, you should also call **tixPushGrab** and **tixPopGrab** in place of the Tk **grab** and **grab release** commands. Otherwise, the behavior of these widgets may be undefined.

KEYWORDS

Tix(n),grab(n)

NAME

Wm - Tix's addition to the standard TK **wm** command.

SYNOPSIS

wm *capture pathName*

wm *release pathName*

DESCRIPTION

The **wm capture** and the **wm release** commands change the toplevel attribute of Tk widgets.

COMMAND OPTIONS

wm capture *pathName*

Converts the toplevel window specified by *pathName* into a non-toplevel widget. Normally this command is called to convert a **toplevel** widget into a **frame** widget. The newly-converted frame widget is un-mapped from the screen. To make it appear inside its parent, you must call a geometry manager (e.g. pack) explictly.

wm release *pathName*

Makes the non-toplevel window specified by *pathName* into a toplevel widget. Normally this command is called to convert a **frame** widget into a **toplevel** widget, but it can also be used on any non-toplevel widget (e.g, button). The newly-converted toplevel window is automatically mapped to the screen. To prevent it from appearing in the screen, you must call **wm withdraw** immediately after calling **wm release**.

KEYWORDS

Tix(n)

NAME

Compound − Create multi-line compound images.

SYNOPSIS

image create compound ?*name*? ?*options*?

DESCRIPTION

Compound image types can be used to create images that consists of multiple horizontal lines; each line is composed of a series of items (texts, bitmaps, images or spaces) arranged from left to right. Compound images are mainly used to embed complex drawings into widgets that support the **−image** option. As shown in the EXAMPLE section below, a compound image can be used to display a bitmap and a text string simutaneously in a TK **button(n)** widget.

CREATING COMPOUND IMAGES

Like all images, compound images are created using the **image create** command. Compound images support the following *options*:

−background *color*
Specifies the background color of the compound image. This color is also used as the default background color for the bitmap items in the compound image.

−borderwidth *pixels*
Specifies a non-negative value indicating the width of the 3-D border drawn around the compound image.

−font *color*
Specifies the default font for the text items in the compound image.

−foreground *color*
Specifies the default foreground color for the bitmap and text items in the compound image.

−padx *value*
Specifies a non-negative value indicating how much extra space to request for the compound image in the X-direction. The *value* may have any of the forms acceptable to **Tk_GetPixels(3)**.

−pady *value*
Specifies a non-negative value indicating how much extra space to request for the compound image in the Y-direction.

−relief *value*
Specifies the 3-D effect desired for the background of the compound image. Acceptable values are **raised**, **sunken**, **flat**, **ridge**, and **groove**.

−showbackground *value*
Specifies whether the background and the 3D borders should be drawn. Must be a valid boolean value. By default the background is not drawn and the compound image appears to have a transparent background.

−window *pathName*
Specifies the window in which the compound image is displayed. One compound image can be displayed in only one window. When that window is destroyed, the compound image is automatically destroyed as well. This option must be specified when calling the **image create compound** command and cannot be changed by the **configure** image command.

IMAGE COMMAND

When a compound image is created, Tk also creates a new command whose name is the same as the image. This command may be used to invoke various operations on the image. It has the following general form:

imageName option ?*arg arg ...*?

Option and the *arg*s determine the exact behavior of the command. The following commands are possible for compound images:

imageName **add line** ?*option value ...*?

Creates a new line at the bottom of the compound image. Lines support the following *options*:

–anchor value

Specifies how the line should be aligned along the horizontal axis. When the values are **w**, **sw** or **nw**, the line is aligned to the left. When the values are **c**, **s** or **n**, the line is aligned to the middle. When the values are **e**, **se** or **ne**, the line is aligned to the right.

–padx *value*

Specifies a non-negative value indicating how much extra space to request for this line in the X-direction.

imageName **add** *item-type* ?*option value ...*?

Creates a new item of the type *item-type* at the end of the last line of the compound image. All types of items support these following common *options*:

-anchor value

Specifies how the item should be aligned along the vertical axis. When the values are **n**, **nw** or **ne**, the item is aligned to the top of the line. When the values are **c**, **w** or **e**, the item is aligned to the middle of the line. When the values are **s**, **se** or **sw**, the item is aligned to the bottom of the line.

–padx *value*

Specifies a non-negative value indicating how much extra space to request for this item in the X-direction.

–pady *value*

Specifies a non-negative value indicating how much extra space to request for this item in the Y-direction.

item-type can be any of the following:

imageName **add bitmap** ?*option value ...*?

Creates a new bitmap item of at the end of the last line of the compound image. Additional *options* accepted by the bitmap type are:

–background *color*

Specifies the background color of the bitmap item.

–bitmap *name*

Specifies a bitmap to display in this item, in any of the forms acceptable to **Tk_GetBitmap(3)**.

–foreground *color*

Specifies the foreground color of the bitmap item.

imageName **add image** ?*option value ...*?

Creates a new image item of at the end of the last line of the compound image. Additional *options* accepted by the image type are:

–image *name*

Specifies an image to display in this item. *name* must have been created with the **image create** command.

imageName **add space** ?*option value ...*?

Creates a new space item of at the end of the last line of the compound image. Space items do not display anything. They just acts as space holders that add additional spaces between items inside a compound image. Additional *options* accepted by the image type are:

−width *value*
Specifies the width of this space. The *value* may have any of the forms acceptable to **Tk_GetPixels(3)**.

−height *value*
Specifies the height of this space. The *value* may have any of the forms acceptable to **Tk_GetPixels(3)**.

imageName **add text** ?*option value ...*?
Creates a new text item of at the end of the last line of the compound image. Additional *options* accepted by the text type are:

−background *color*
Specifies the background color of the text item.

−font *name*
Specifies the font to be used for this text item.

−foreground *color*
Specifies the foreground color of the text item.

−justify *value*
When there are multiple lines of text displayed in a text item, this option determines how the lines line up with each other. *value* must be one of **left**, **center**, or **right**. **Left** means that the lines' left edges all line up, **center** means that the lines' centers are aligned, and **right** means that the lines' right edges line up.

−text *string*
Specifies a text string to display in this text item.

−underline *value*
Specifies the integer index of a character to underline in the text item. 0 corresponds to the first character of the text displayed in the text item, 1 to the next character, and so on.

−wraplength *value*
This option specifies the maximum line length of the label string on this text item. If the line length of the label string exceeds this length, it is wrapped onto the next line, so that no line is longer than the specified length. The value may be specified in any of the standard forms for screen distances. If this value is less than or equal to 0 then no wrapping is done: lines will break only at newline characters in the text.

imageName **cget** *option*
Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the **image create compound** command.

imageName **configure** ?*option*? ?*value option value ...*?
Query or modify the configuration options for the image. If no *option* is specified, returns a list describing all of the available options for *imageName* (see **Tk_ConfigureInfo** for information on the format of this list). If *option* is specified with no *value*, then the command returns a list describing the one named option (this list will be identical to the corresponding sublist of the value returned if no *option* is specified). If one or more *option−value* pairs are specified, then the command modifies the given option(s) to have the given value(s); in this case the command returns an empty string. *Option* may have any of the values accepted by the **image create compound** command, except the **−window** option

EXAMPLE

The following example creates a compound image with a bitmap and a text string and places this image into a Tk **bitton(n)** widget. Notice that the image must be created after the creation of the window that it resides in.

```
button .b
```

```
set img [image create compound -window .b]
$img add line
$img add bitmap -bitmap warning
$img add space -width 8
$img add text -text "Warning" -underline 0
.b config -image $img
pack .b
```

KEYWORDS

image(n), Tix(n)

NAME

pixmap − Create color images from XPM files.

SYNOPSIS

image create pixmap ?*name*? ?*options*?

DESCRIPTION

XPM is a popular X Window image file format for storing color icons. The **pixmap** image type defined by the **Tix(n)** library can be used to create color images using XPM files.

CREATING PIXMAPS

Like all images, pixmaps are created using the **image create** command. Pixmaps support the following *options*:

−data *string*

Specifies the contents of the source pixmap as a string. The string must adhere to the XPM file format (e.g., as generated by the **pixmap(1)** program). If both the **−data** and **−file** options are specified, the **−data** option takes precedence. Please note that the XPM file parsing code in the xpm library is extremely fragile. The first line of the string must be "**/* XPM */**" or otherwise a segmatation fault will be caused.

−file *name*

name gives the name of a file whose contents define the source pixmap. The file must adhere to the XPM file format (e.g., as generated by the **pixmap(1)** program).

IMAGE COMMAND

When a pixmap image is created, Tk also creates a new command whose name is the same as the image. This command may be used to invoke various operations on the image. It has the following general form:

imageName option ?*arg arg ...*?

Option and the *arg*s determine the exact behavior of the command. The following commands are possible for pixmap images:

imageName **cget** *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the **image create pixmap** command.

imageName **configure** ?*option*? ?*value option value ...*?

Query or modify the configuration options for the image. If no *option* is specified, returns a list describing all of the available options for *imageName* (see **Tk_ConfigureInfo** for information on the format of this list). If *option* is specified with no *value*, then the command returns a list describing the one named option (this list will be identical to the corresponding sublist of the value returned if no *option* is specified). If one or more *option−value* pairs are specified, then the command modifies the given option(s) to have the given value(s); in this case the command returns an empty string. *Option* may have any of the values accepted by the **image create pixmap** command.

KEYWORDS

pixmap(1), image(n), Tix(n)

NAME

tix − Manipulate Tix internal state

SYNOPSIS

tix *option* ?*arg arg ...*?

CONFIGURATION OPTIONS

The Tix application context supports the following configuration options. Usually, these options are set using the X resource database, i.e., in the user's **.Xdefault** file. For example, to choose a different color scheme for the Tix widgets, these two lines can be added to the user's **.Xdefault** file:

```
*TixScheme:     Gray
*TixFontSet:    14Point
```

Name: **binding**
Class: **Binding**
Switch: **−binding**

This is an obsolete option.

Name: **debug**
Class: **Debug**
Switch: **−debug**

Specifies whether the Tix widgets should run in debug mode.

Name: **tixFontSet**
Class: **TixFontSet**
Switch: **−fontset**

Specifies the fontset to use for the Tix widgets. Valid options are **TK**, **12Point** and **14Point**. **TK** specifies that the standard TK fonts should be used. The default value is **14Point**.

Name: **tixScheme**
Class: **TixScheme**
Switch: **−scheme**

Specifies the color scheme to use for the Tix widgets. Valid options are **TK**, **Gray**, **Blue**, **Bisque**, **SGIGray** and **TixGray**. The default value is **TixGray**. If you want the standard TK color scheme, you can use the value **TK**. If you want to use the TK 3.6 bisque color scheme, you can use the value **Bisque**.

Name: **tixSchemePriority**
Class: **TixSchemePriority**
Switch: **−schemepriority**

Specifies the priority level of the TK options set by th Tix schemes. Please refer to the TK **option(n)** manual page for a discussion of the priority level of Tix options. The default value is 79, which makes the Tix schemes at a higher priority than the settings in the .Xdefaults file. If you want to allow the Tix schemes to be overridden by the settings in the .Xdefaults file, you can set the following line in you .Xdefaults file:
*TixSchemePriority: 21

DESCRIPTION

The **tix** command provides access to miscellaneous elements of Tix's internal state and the Tix **application context**. Most of the information manipulated by this command pertains to the application as a whole, or to a screen or display, rather than to a particular window. The command can take any of a number of different forms depending on the *option* argument. The legal forms are:

pathName **cget** *option*

Returns the current value of the configuration option given by *option*. *Option* may be any of the options described in the **CONFIGURATION OPTIONS** section.

tix configure ?*option*? *?value option value ...*?

Query or modify the configuration options of the Tix application context. If no *option* is specified, returns a list describing all of the available options (see **Tk_ConfigureInfo** for information on the format of this list). If *option* is specified with no *value*, then the command returns a list describing the one named option (this list will be identical to the corresponding sublist of the value returned if no *option* is specified). If one or more *option−value* pairs are specified, then the command modifies the given option(s) to have the given value(s); in this case the command returns an empty string. *Option* may be any of the options described in the **CONFIGURATION OPTIONS** section.

tix filedialog ?*class*?

Returns the file selection dialog that may be shared among different modules of this application. This command will create a file selection dialog widget when it is called the first time. This dialog will be returned by all subsequent calls to **tix filedialog**. An optional *class* parameter can be passed to specified what type of file selection dialog widget is desired. Possible options are **tixFileSelectDialog** or **tixExFileSelectDialog.**

tix option ?*args ...*?

Manipulates the options manitained by the Tix scheme mechanism. Available options are:

```
active_bg          active_fg          bg
bold_font          dark1_bg           dark1_fg
dark2_bg           dark2_fg           disabled_fg
fg                 fixed_font         font
inactive_bg        inactive_fg        input1_bg
input2_bg          italic_font        light1_bg
light1_fg          light2_bg          light2_fg
menu_font          output1_bg         output2_bg
select_bg          select_fg          selector
```

The arguments to the **tix option** command can take the following form(s):

tix option get *option*

Returns the current value of *option*.

tix resetoptions *newScheme newFontSet* ?*newScmPrio*?

Resets the scheme and fontset of the Tix application to *newScheme* and *newFontSet*, respectively. This affects only those widgets created **after** this call. Therefore, it is best to call the **resetoptions** command **before** the creation of any widgets in a Tix application.

The optional parameter *newScmPrio* can be given to reset the priority level of the TK options set by the Tix schemes.

BUGS

Because of the way TK handles the X option database, after tixwish has started up, it is not possible to reset the color schemes and font sets using the **tix config** command. Instead, the **tix resetoptions** command must be used.

The tk_setPalette command does not work very well under Tix. To use it, one must follow these steps:

```
tix resetoptions TK TK
tk_setPalette lightblue
```

KEYWORDS

file selection dialog