EDOC224

Midas

TK Support Library (common.tk.tcl)

Edition 3.1

October 1997

Nuclear Physics Support Group

Central Laboratory of the Research Councils
Daresbury Laboratory

TK Support Library

This document describes a Support Library (SL) of TCL procedures which can be used to create a graphical interface (GUI) for a TCL application.

The procedure interface is based very strongly on the Xview interface (xv) originally created by David Brightly and used for the creation of the GUI used by Eurogam. This support library was initially created with the view to ease the transition of the Eurogam GUI from use of Xview (and thus dependence on Sun MicroSystems) to the use of the platform independent package TK which generates a Motif compliant look and feel while requiring only X11 support from the Unix system it is running in. The use of the Support Library however makes generation of a GUI using TK very much quicker and easier than direct use of the “raw” TK commands and thus its use has many advantages other than in its original role of conversion of existing Eurogam applications.

Throughout this document the word **widget** is used to mean a graphical construction created by one of the SL procedures. A **widget** consists of a TK frame window holding a number of TK windows. The fact that a **widget** is a composite of elemental TK windows is one of the reasons why using the SL procedures makes generation of GUIs quicker and easier. In all cases several TK commands are required to implement the creation of a **widget**. By convention **widget** names are in **upper case**. While use of this convention is not essential when using most of the SL procedures it is assumed by the call-back mechanism and the **click** procedure.

The SL procedures require that a procedure named **click** is supplied to handled all call backs from the **widgets** created for which a TCL procedure to be invoked is not supplied when the **widget** is created. The **click** procedure has a single argument which is a TCL list. The list elements are

index 0	frame name containing the **widget**
index 1	command
index 2	the **widget** name
index 3	value
index 4	index

The contents of *command* and *value* are dependent on the **widget** type. *command* is used mostly for special purposes - for example if a window is **QUIT** via the window manager then the application is informed by invoking the **click** procedure with *command* having the value **:quit**. The *value* and *index* elements are not present for all **widget** types.
Where a TCL procedure to be invoked when an event occurs is supplied to the **widget** creation procedure then the procedure is invoked during the call back with a single list argument as above but omitting the first 3 items. This may mean (for example the **button** widget) that there is no argument .

The library contains a default version of the **click** procedure which is described later in this document.

widget creation procedures

All **widget** creation procedures have the same format:

procedure-name widget-name [argument keyword value]

There may be a variable number of keyword/value pairs (including none).
For further details of argument values the man pages for TK should be consulted.

procedure

:frame *name*

Creates the primary frame for a TCL interpreter or a secondary top level frame for the interpreter.
name is used to generate the path name of the window to be created. It may have either the format of a widget name (e.g. :frame SYS) or the format required by TK for a path name (e.g. :frame .sys).
If the supplied frame name does not start with a period it is assumed to be a widget name. The name will be lower-cased (using the *string tolower* command) and a period prefixed. If the name thus generated is the same as the name returned by the *self frame-name* command (see common.tcl) then it is assumed that this is the primary frame for the interpreter (see also the *:toplevel* argument following).
The conditions for this to be the primary frame will be met if *name* is the same as the global TCL variable *frame*.

bindings

A number of bindings are made by default by the **:frame** procedure.

(1) when the window manager *delete window* event is generated (normally by the frame being **Quit** by the user) then the TCL command

click NAME :quit

will be executed where *NAME* is the widget name generated from *name*.

(2) If the frame being created is the primary frame for the interpreter then a binding is made applicable to all windows in the interpreter for the keyboard sequence **Ctrl**+**Alt**+**Delete** to the sequence required to orderly finish and destroy the entire application. Since interpreters are automatically created as necessary if they do not exist when invoked this is a convenient way of causing a specific interpreter to be reloaded. The sequences **Ctrl+Delete** and **Alt+Delete** may also be used which is particularly of use when using terminals for which the full sequence has more dramatic effects.

(3) If the frame being created is the primary frame for the interpreter then bindings are made applicable to all windows in the interpreter for the keyboard sequences **F1** , **Shift+F1** and **Alt+F1**. These invoke features in the

online Help system. For further details see the section later in this document which describes the Help system.

(4) For frames created with the parameter **:transient T** a binding is made such that the window is destroyed when an **Unmap** event occurs (normally by the frame being **Closed** by the user).

(5) A binding is made such that if button 1 is pressed while positioned over the frame footer area (if **:show-footer T**) then the footer is cleared.

(6) A binding is made such that if button 3 is pressed while positioned over the frame footer area (if **:show-footer T**) then the frame busy indicator is cleared.

(7) For frames (other than those created with the parameter **:transient T)** a binding is made such that when an **Unmap** event occurs the global variable **WMapState** is set to 0 and the procedure **click-unmap** is invoked if it exists.

(8) For frames (other than those created with the parameter **:transient T)** a binding is made such that when a **Map** event occurs the global variable **WMapState** is set to 1 and the procedure **click-map** is invoked if it exists.

(9) For frames (other than those created with the parameter **:transient T)** a binding is made such that when a **Visibility** event occurs the procedure **click-visibility** is invoked if it exists.

(10) If the frame being created is the primary frame for the interpreter then bindings are made applicable to all windows in the interpreter for the keyboard sequence **Shift+F2**. This starts a dialogue which enables the TCL source files used by the interpreter to be displayed.

arguments

The **:frame** procedure may have a number of arguments however none of these is mandatory.

:label *string*

string is used as the title of the window and will appear centered in the top bar of the window bounding box.

e.g. :label "System Control"

The default value for the **:label** argument is "*name* frame".

:icon-name *string*

string is used as the title of the icon for the window.

e.g :icon-name "SysCntl"

The default value for the **:icon-name** argument is "*name".*

:icon-image *string*

string is the name of a file which should contain a X bit map to be used as the icon for the window.

e.g :icon-image “@/MIDAS/tcl/bitmaps/egarray”

The default value for the **:icon-image** argument is “@$env(ICONHOME)/default”.

:w *number*

number is the width of the window in pixels.

e.g :w 600

:h *number*

number is the height of the window in pixels.

e.g. :h 400

The **:w** and **:h** parameters may be omitted in which case the packer geometry manager will be used and the size of the window will be determined by its contents. If however the **:w** and **:h** parameters are supplied then the placer geometry manager will be used by default for this window.

:toplevel **T|F**

If the value of this argument is **T** (true) then the window is created using the TK *toplevel* command otherwise it is created using the TK *frame* command.

By default (if this argument is not supplied) the *frame* command is used if the name supplied for the frame is the same as the value returned by the *self frame-name* command (the global variable frame) otherwise the *toplevel* command is used.

:transient **T**

If the value of this argument is **T** (true) then the window is taken to only have a short term existence. A binding is made for the window which causes it to be removed from the screen (destroyed) if the window is closed (unmapped). A global variable is created by taking the last component of the window name and appending the string *visible.* (i.e. the name *.tape.ss* would create the variable *ssvisible*. This variable has the value 1 when the transient window is created and the value 0 when it is destroyed.

The default value for this option is F (false). A transient window is never iconized.

:show-footer **T**

If the value of this argument is **T** (true) then a footer is created for this window.. The footer occupies an area of one character height at the bottom of the window. The footer is described in more detail later in this document.

The default is that no footer is created

:resize **F**

If the value of this argument is **F** (false) then the window created may not be resized. The initial size of the window is determined by the geometry manager (if packing is used) or by the **:w** and **:h** parameter values if specified and placing is used.

By default windows may be resized using the mouse

:closed **T**

If the value of this argument is **T** (true) then the window is created and iconified.

By default windows are created open.

:pack **T**

If the value of this argument is **T** (true) then the packer geometry manager is used to position the footer (if it exists) at the bottom left hand corner of the window.

:place **T**

If the value of this argument is **T** (true) then the placer geometry manager is used to position the footer (if it exists) at the bottom left hand corner of the window.

If neither the **:pack** nor **:place** arguments are specified then the choice of geometry manager is determined by the presence (place) or absence (pack) of the **:w** and **:h** arguments.

procedure

:panel *name*

Creates a secondary frame within the current top level frame for the TCL interpreter.
name is used to generate the path name of the window to be created. It should have the format of a widget name (e.g. :panel P) and should be unique within this interpreter. The name will be lower-cased (using the *string tolower* command) in order to generate the path name used to create the window.
The path name generated has the form *.framename.panelname.*

The **:panel** procedure may have a number of arguments however none of these is mandatory.

arguments

:x *number*

number is the x-co-ordinate within the current frame of the panel window in pixels
e.g. :x 400

:y *number*

number is the y-co-ordinate within the current frame of the panel window in pixels
e.g. :y 200

The **:x** and **:y** parameters may be omitted in which case the packer geometry manager will be used and the position of the window will be determined by the contents of the master (frame) window. If however the **:x** and **:y** parameters are supplied then the placer geometry manager will be used to position the window within the master (frame) window.
Warning: Note that it is not possible to mix use of the packer and placer geometry managers within the same window. Thus all panels must request or imply that the same geometry manager is used.

:w *number*

number is the width of the window in pixels.
e.g. :w 200

:h *number*

number is the height of the window in pixels.
e.g. :h 60

The **:w** and **:h** parameters may be omitted in which case the size of the window will be determined by its contents. If however the **:w** and **:h** parameters are supplied then these will be used to **configure** the panel window.

:anchor *where*

where should be a valid window point to be used when placing or packing the panel window within the frame window. The parameter supplied is made the default to be used for any windows created within the panel.
Valid points are n,s,e,w,nw,ne,se,sw The default anchor point is **nw**.

:layout **L|H|T|V|R|B**

Determines which side of the master (frame) this panel window should be packed. **L** and **H** pack to side left; **T** and **V** pack to side top; **R** packs to side right and **B** packs to side bottom. If **R** (right) is specified then the anchor point is set to **ne** by default. The layout parameter is used for the panel window only if the window is to be packed within the frame. That is only if **:x** and **:y** are **not** specified. However the parameter supplied is made the default to be used for any windows created within the panel.
The default is that **layout left** is used

:expand *option*

option should be any valid value for the -expand configuration option to be used by the packer geometry manager. The parameter supplied is made the default to be used for any windows created within the panel.
The default is the **expand yes** is used.

:fill *option*

option should be any valid value for the -fill configuration option to be used by the packer geometry manager. The parameter supplied is made the default to be used for any windows created within the panel.
The default is the **fill both** is used.

:ipadx *value*

value should be any valid value for the -ipadx configuration option to be used by the packer geometry manager. The parameter supplied is made the default to be used for any windows created within the panel.
The default is the **ipadx 0** is used.

:ipady *value*

value should be any valid value for the -ipady configuration option to be used by the packer geometry manager. The parameter supplied is made the default to be used for any windows created within the panel.
The default is the **ipady 0** is used.

:padx *value*

value should be any valid value for the -padx configuration option to be used by the packer geometry manager. The parameter supplied is made the default to be used for any windows created within the panel.
The default is the **padx 0** is used.

:pady *value*

value should be any valid value for the -pady configuration option to be used by the packer geometry manager. The parameter supplied is made the default to be used for any windows created within the panel.
The default is the **pady 0** is used.

:row-gap *number*

number is the additional space (in pixels) to be allowed in addition to one character height between rows of windows created within the panel window (see **:next-row** procedure). The placer geometry manager is selected to be used for all widgets created within the panel window.
By default row-gap is set to 0 and the geometry manager selected is the packer

:col-gap *number*

number is the additional space (in pixels) to be allowed in addition to one character width between columns of windows created within the panel window (see **:next-col** procedure). The placer geometry manager is selected to be used for all widgets created within the panel window.
By default col-gap is set to 0 and the geometry manager selected is the packer.

See also common procedure arguments defined later in this document.

procedure

:button *name*

Creates a button widget within the current panel frame for the TCL interpreter. *name* is used to generate the path name of the window to be created. It should have the format of a widget name (e.g. :button BUT) and should be unique within this interpreter. The name will be lower-cased (using the *string tolower* command) in order to generate the path name used to create the window.

When the button is selected using the mouse then ,if available, the TCL script supplied using the **:command** parameter will be executed. If this is not available the TCL command

click FRAME " " BUTTON

will be executed where **FRAME** and **BUTTON** are widget names.
The path name generated has the form *.framename.panelname.buttonname.*
The :button procedure may also be used directly within a frame window (i.e. before any panel window has been created).

The **:button** procedure may have a number of arguments however none of these is mandatory.

arguments

:x *number*

number is the x-co-ordinate within the current panel of the button window in pixels

e.g. :x 40

:y *number*

number is the y-co-ordinate within the current panel of the button window in pixels

e.g. :y 20

The **:x** and **:y** parameters may be omitted in which case the packer geometry manager will be used and the position of the window will be determined by the contents of the master (panel) window. If however the **:x** and **:y** parameters are supplied then the placer geometry manager will be used to position the window within the master (panel) window.
Warning: Note that it is not possible to mix use of the packer and placer geometry managers within the same window. Thus all windows within a panel must request or imply that the same geometry manager is used.

:w *number*

number is the width of the button in characters.

e.g. :w 20

For compatibility with other widget procedures the argument :**valuex** is recognised and treated as equivalent to **:w**.

:h *number*

number is the height of the button in lines of text.

e.g. :h 2

The **:w** and/or **:h** parameters may be omitted in which case the size of the button will be determined by its contents. If however the **:w** and **:h** parameters are supplied then these will be used to **configure** the button window. Normally **:h** will not be supplied.

:label *string*

label is a text string to be used to configure the button and will be displayed within the button widget.

The default is the name of the button widget. Normally the string supplied is text to be written onto the button widget. However if the string supplied starts with the character @ then it is assumed to be a bitmap file descriptor.

:padx *value*

value should be any valid value for the -padx configuration option to be used by the TCL button command.

The default is that **padx 0** is used.

:pady *value*

value should be any valid value for the -pady configuration option to be used by the TCL button command.

The default is that **pady 0** is used.

:hlt *value*

value should be any valid value for the -highlightthickness configuration option to be used by the TCL button command.

The default is that **hlt 1** is used.

:colour *value*

value should be any valid colour specification and will be used to configure the background component of the button window. This overrides any colour selection made using the **:colour** command or any user colour selection.

It is recommended that this feature should be used only where important since the *value* supplied may be incompatible with the current user colour selection.

:command *string*

string is any TCL command or commands to be executed when the button is selected.

If a command is supplied and executed in this manner then the **click** procedure is not invoked.

See also common procedure arguments defined later in this document.

procedure

:text *name*

Creates a text widget within the current panel frame for the TCL interpreter. A text widget is a frame window containing a label window and an entry window. *name* is used to generate the path name of the windows to be created. It should have the format of a widget name (e.g. :text TXT) and should be unique within this interpreter. The name will be lower-cased (using the *string tolower* command) in order to generate the path name used to create the windows.
When the return key is pressed while the keyboard focus is on the entry window then ,if available, the TCL script supplied using the **:command** parameter will be executed. If this is not available the TCL command

click FRAME " " TEXT text-string

will be executed where **FRAME** and **TEXT** are widget names and **text-string** is the information entered in the entry window.
The path names generated have the root *framename.panelname.textname* and the items *label* and *entry.*
The **:text** procedure may also be used directly within a frame window (i.e. before any panel window has been created).

Normally it is necessary when typing information into the entry window to press the *return* key in order to generate an event and cause the procedure linked to the entry to be executed. Often users fail to do this and so a binding is made to the mouse focus and the contents of the entry window are checked whenever the mouse focus leaves the entry window. If information is found to have been typed into the entry window which has not been seen by a *return* key event then an action is taken as is the *return* key had been pressed.

The **:text** procedure may have a number of arguments however none of these is manditory.

arguments

:x *number*

number is the x-co-ordinate within the current panel of the text widget in pixels

e.g. :x 40

:y *number*

number is the y-co-ordinate within the current panel of the text widget in pixels

e.g. :y 20

The **:x** and **:y** parameters may be omitted in which case the packer geometry manager will be used and the position of the window will be determined by the contents of the master (panel) window. If however the **:x** and **:y** parameters are supplied then the placer geometry manager will be used to position the window within the master (panel) window.
Warning: Note that it is not possible to mix use of the packer and placer geometry managers within the same window. Thus all windows within a panel must request or imply that the same geometry manager is used.

:w *number*

number is the width of the text entry window in characters and is also used to calculate the width of the text widget..

e.g. :w 20

:h *number*

number is the height of the text widget in pixels.

e.g. :h 20

The **:w** and/or **:h** parameters may be omitted in which case the size of the text widget will be determined by its contents. If however the **:w** and **:h** parameters are supplied then these will be used to **configure** the text window. Normally **:h** will not be supplied.

:valuex *value*

value is the position of the left hand edge of the entry window with relation to the left hand edge of the text widget. It is thus the space available to write the label string but also allows alignment of the entry windows of several text widgets. The value may be expressed in units of pixels or characters.

e.g. :valuex 100 (pixels); :valuex 15c (characters)

:label *string*

string is a text string to be written into the label component of the text widget.

The default is the name of the button widget.

:**value** *string*

string is a text string to be written initially into the entry component of the text widget.

The default is a null string.

:shifters **T|F**

If the option **T** is supplied then small buttons are placed at each end of the entry widget which can be used to scroll the contents of the entry right or left in the case when the width of the contents of the entry is greater than the physical width of the entry widget.

The default is that these buttons are not drawn.

:**read-only** **T|F**

This is an alias for **:deaf** supplied for compatibility with existing scripts.

:**size** **S|N|L**

Specifies the size of the font to be used for characters in the label and entry windows. Options available are small (**S**); normal (**N**) and large (**L**).

The default is that size N (normal) is used.

:command *string*

string is any TCL command or commands to be executed when the keyboard return key is pressed.

If a command is supplied and executed in this manner then the **click** procedure is not invoked.

See also common procedure arguments defined later in this document.

procedure

:list *name*

Creates a list widget within the current panel frame for the TCL interpreter. A list widget is a frame window containing a scrollbar window and a listbox window.
name is used to generate the path name of the windows to be created. It should have the format of a widget name (e.g. :list LST) and should be unique within this interpreter. The name will be lower-cased (using the *string tolower* command) in order to generate the path name used to create the windows.
When the mouse button 1 is released while positioned over the listbox window then ,if available, the TCL script supplied using the **:command** parameter will be executed. If this is not available the TCL command

click FRAME :list-select LIST text-string selection

will be executed where **FRAME** and **LIST** are widget names. If the option **exclusive T** is active **text-string** is the information contained in the currently selected item of the list and **selection** is the index number for the currently selected item. If the option **exclusive F** is active **text-string** is a TCL list containing information for each of the currently selected items and **selection** is a TCL list of the index numbers for each of the currently selected items.
The path names generated have the root *framename.panelname.listname* and the items *scroll* and *list.*
The **:list** procedure may also be used directly within a frame window (i.e. before any panel window has been created).

A binding is made to the keyboard *Control + P* sequence which causes the contents of the listbox window to be sent to the printer. A binding is made to the keyboard *Control + S* sequence which causes the contents of the listbox window to be saved to file.

The **:list** procedure may have a number of arguments however none of these is mandatory.

arguments

:x *number*

number is the x-co-ordinate within the current panel of the list widget in pixels

e.g. :x 40

:y *number*

number is the y-co-ordinate within the current panel of the list widget in pixels

e.g. :y 20

The **:x** and **:y** parameters may be omitted in which case the packer geometry manager will be used and the position of the window will be determined by the contents of the master (panel) window. If however the **:x** and **:y** parameters are supplied then the placer geometry manager will be used to position the window within the master (panel) window.
Warning: Note that it is not possible to mix use of the packer and placer geometry managers within the same window. Thus all windows within a panel must request or imply that the same geometry manager is used.

:w *number*

number is the width of the list widget in pixels. If **:w** is not defined then the width of the list widget is calculated using the value supplied for **:list-width**.

e.g. :w 20

:h *number*

number is the height of the list widget in pixels and is also the height of the list and scroll windows.

e.g. :h 20

:rows *number*

number is the number of rows of text to be contained by the listbox window. This is used to define the height of the list widget (see **:h**).

:cols *number*

number is the number of columns of text to be contained by the listbox window. This is used to define the width of the listbox window and is used to position the scrollbar window (see **:w** and **:list-width**).

:list-width *number*

number is the width of the listbox window in pixels and is used to position the scrollbar window. It is also used to determine the width of the list widget if **:w** is not defined.

There are obvious interactions between **:w**, **:h**, **:rows**, :**cols** and **:list-width** and so it is not sensible or necessary to define all of these.

The preferred choice is to specify the size of the list using **:rows** and **:cols** since these are specified using units of characters and are thus not dependent on the font size selected.

:strings *list*

list is a TCL list of text strings which are to be written into the listbox component of the list widget.

The default is that the listbox is left empty.

:**exclusive** **T|F**

If the option **T** is selected then only one item at a time is selected from the list of strings defined by the **:strings** argument. Selecting an item from the list automatically causes any currently selected item to be deselected. If the option **F** is selected then any number of items from the list may be selected concurrently.

The default option is **exclusive T**.

:browsing **T|F**

This argument is treated as an alias for **:exclusive**.

:see *number*

Defines an element of the list which must be visible in the listbox widget. To ensure that the beginning of the list is visible then **:see 0** should be specified.

The default is that the start of the list is visible; i.e. **:see 0.** For compatibility with use in other commands **:show** is recognised as an alias for **:see**.

:scroll **V|H|B|N**

Specifies the generation and location of scroll-bar windows. These may be omitted (**N** - none); be drawn only vertically to the right of the listbox window (**V** - vertical); be drawn only horizontally below the listbox window (**H** - horizontal) or be drawn both vertically and horizontally (**B** - both).

:command *string*

string is any TCL command or commands to be executed when the list is selected.

If a command is supplied and executed in this manner then the **click** procedure is not invoked.

See also common procedure arguments defined later in this document.

procedure

:galley *name*

Creates a galley widget within the current panel frame for the TCL interpreter. A galley widget is a frame window containing a scrollbar window and a text window.
name is used to generate the path name of the windows to be created. It should have the format of a widget name (e.g. :galley GALY) and should be unique within this interpreter. The name will be lower-cased (using the *string tolower* command) in order to generate the path name used to create the windows.
A galley widget is non interactive (apart from the scroll bar) and thus no **click** procedure is invoked.
The **:galley** procedure may also be used directly within a frame window (i.e. before any panel window has been created).

A binding is made to the keyboard *Control + P* sequence which causes the contents of the text window to be sent to the printer. A binding is made to the keyboard *Control + S* sequence which causes the contents of the text window to be saved to file.

The **:galley** procedure may have a number of arguments however none of these is mandatory.

arguments

:x *number*

number is the x-co-ordinate within the current panel of the galley widget in pixels

:y *number*

number is the y-co-ordinate within the current panel of the galley widget in pixels
The **:x** and **:y** parameters may be omitted in which case the packer geometry manager will be used and the position of the window will be determined by the contents of the master (panel) window. If however the **:x** and **:y** parameters are supplied then the placer geometry manager will be used to position the window within the master (panel) window.
Warning: Note that it is not possible to mix use of the packer and placer geometry managers within the same window. Thus all windows within a panel must request or imply that the same geometry manager is used.

:rows *number*

number is the number of rows of text to be contained by the text window. This is used to define the height of the text and scrollbar windows.

:cols *number*

number is the number of columns of text to be contained by the text window. This is used to define the width of the text window and is used to position the scrollbar window.

:strings *string*

string is any text string which is to be written into the text component of the galley widget. It may contain any characters acceptable to the text window. Lines of text longer than the width of the text window will be wrapped at word boundaries.

The default is that the text is left empty.

:see *number*

Makes the first character on the line after the one given by *number* visible at the top of the window.

By default when text is written into the window the last line written is left visible at the bottom of the window. For compatibility with use in other commands **:show** is recognised as an alias for **:see**.

:size **S|N|L**

Specifies the size of the font to be used for characters in the text window. Options available are small (**S**); normal (**N**) and large (**L**).

The default is that size N (normal) is used.

:scroll **V|H|B|N**

Specifies the generation and location of scroll-bar windows. These may be omitted (**N** - none); be drawn only vertically to the right of the text window (**V** - vertical); be drawn only horizontally below the text window (**H** - horizontal) or be drawn both vertically and horizontally (**B** - both).

The default option is V.

See also common procedure arguments defined later in this document.

procedure

:message *name*

Creates a message widget within the current panel frame for the TCL interpreter. A message widget is a frame window containing a message window.

name is used to generate the path name of the window to be created. It should have the format of a widget name (e.g. :message MSG) and should be unique within this interpreter. The name will be lower-cased (using the *string tolower* command) in order to generate the path name used to create the windows.

A message widget is none interactive and thus no **click** procedure is invoked.

The path names generated have the root *framename.panelname.message-name* and the item *msg.*

The **:message** procedure may also be used directly within a frame window (i.e. before any panel window has been created).

The message window is placed within its own frame window to create a message widget in order to enable correct implementation of the :valuex argument and also to allow full access to options such as the background colour for the message text.

The **:message** procedure may have a number of arguments however none of these is mandatory.

arguments

:x *number*

number is the x-co-ordinate within the current panel of the message widget in pixels

:y *number*

number is the y-co-ordinate within the current panel of the message widget in pixels

The **:x** and **:y** parameters may be omitted in which case the packer geometry manager will be used and the position of the window will be determined by the contents of the master (panel) window. If however the **:x** and **:y** parameters are supplied then the placer geometry manager will be used to position the window within the master (panel) window.

Warning: Note that it is not possible to mix use of the packer and placer geometry managers within the same window. Thus all windows within a panel must request or imply that the same geometry manager is used.

:w *number*

number is the width of the message window in characters.

The **:w** parameter may be omitted in which case the size of the message widget will be determined by its contents.

:valuex *value*

value is the position of the left hand edge of the message in relation to the left hand edge of the message widget. The value may be expressed in units of pixels or characters.

e.g. :valuex 100 (pixels); :valuex 15c (characters)

The **:valuex** parameter may be omitted in which case the size of the message widget will be determined by its contents.

:label *string*

string is a text string to be written into the message widget.

The default is a null string.

:font *string*

string is the specification of the font to be used when writing text into the message widget.

By default the font specified for message widgets will be used. If the **:font** argument is supplied then **:bold** will be ignored

:justify *string*

string is the specification of the justification to be used when writing text into the message widget.

If not supplied the default justification for message windows will be used.

:colour *string*

string is the colour to be used when writing text into the message widget.

By default the colour specified for message widgets will be used.

:anchor *where*

where can be any valid window point to be used when placing the message within its window.

:anchor is only meaningful when **:valuex** has also been supplied otherwise it is ignored and the message packed to the left of its window. Valid points are n,s,e,w,nw,ne,se,sw. The default anchor point is **ne** - i.e. the message is placed justified to the right of its window.

See also common procedure arguments defined later in this document.

procedure

:checkbox *name*

Creates a checkbox widget within the current panel frame for the TCL interpreter. A checkbox widget is a frame window containing a label window and a checkbutton window.
name is used to generate the path name of the windows to be created. It should have the format of a widget name (e.g. :checkbox CHK) and should be unique within this interpreter. The name will be lower-cased (using the *string tolower* command) in order to generate the path name used to create the windows.
When the checkbox is selected using the mouse then the TCL command

click FRAME " " CHECKBOX value

will be executed where **FRAME** and **CHECKBOX** are widget names and **value** is the current state (0,1); (off,on) of the checkbox.
The path names generated have the root *framename.panelname.checkbox-name* and the items *label* and *button.*
The **:checkbox** procedure may also be used directly within a frame window (i.e. before any panel window has been created).

The **:checkbox** procedure may have a number of arguments however none of these is mandatory.

arguments

:x *number*

number is the x-co-ordinate within the current panel of the checkbox widget in pixels

e.g. :x 400

:y *number*

number is the y-co-ordinate within the current panel of the checkbox widget in pixels

e.g. :y 200

The **:x** and **:y** parameters may be omitted in which case the packer geometry manager will be used and the position of the window will be determined by the contents of the master (panel) window. If however the **:x** and **:y** parameters are supplied then the placer geometry manager will be used to position the window within the master (panel) window.
Warning: Note that it is not possible to mix use of the packer and placer geometry managers within the same window. Thus all windows within a panel must request or imply that the same geometry manager is used.

:value *value*

value is the initial state of the checkbox (0/off; 1/on).
The default value is 0/off.

:valuex *value*

value is the position of the left hand edge of the checkbutton window with relation to the left hand edge of the checkbox widget. It is thus the space available to write the label string but also allows alignment of the checkbutton windows of several checkbox widgets. The value may be expressed in units of pixels or characters.

e.g. :valuex 100 (pixels); :valuex 15c (characters)

:label *string*

string is a text string to be written into the label component of the checkbox widget.

The default is the name of the checkbox widget.

:w *value*

value is the width of the checkbutton in pixels.

By default the size of the checkbutton is determined by the image written into it.

:h *value*

value is the height of the checkbutton in pixels.

By default the size of the checkbutton is determined by the image written into it.

:colours *list*

list may consist of one or two items which should be any valid colour specification. The first item defines the background colour for the checkbox window and the second item defines the colour when the window is selected.

These colours override any colour selection made using the **:colour** command or any user colour selection.

If the list contains only one item then only the background colour is defined and the normal method is used to define the colour of the window when selected.

See also common procedure arguments defined later in this document.

procedure

:number *name*

Creates a number widget within the current panel frame for the TCL interpreter. A number widget is a frame window containing a label window, entry window and two checkbox windows.
name is used to generate the path name of the windows to be created. It should have the format of a widget name (e.g. :number NUM) and should be unique within this interpreter. The name will be lower-cased (using the *string tolower* command) in order to generate the path name used to create the windows.
When a number is entered either by pressing the Return key while the keyboard focus is on the entry window or by selection of one of the two checkbox windows using the mouse then the TCL command

click FRAME " " NUMBER value

will be executed where **FRAME** and **NUMBE**R are widget names and **value** is the current value of the number widget.
The path names generated have the root *framename.panelname.number-name* and the items *label, entry, button-up* and *button-dn.*
The **:number** procedure may also be used directly within a frame window (i.e. before any panel window has been created).

Normally it is necessary when typing information into the entry window to press the *return* key in order to generate an event and cause the procedure linked to the entry to be executed. Often users fail to do this and so a binding is made to the mouse focus and the contents of the entry window are checked whenever the mouse focus leaves the entry window. If information is found to have been typed into the entry window which has not been seen by a *return* key event then an action is taken as is the *return* key had been pressed.

The **:number** procedure may have a number of arguments however none of these is mandatory.

arguments

:x *number*

number is the x-co-ordinate within the current panel of the number widget in pixels

e.g. :x 400

:y *number*

number is the y-co-ordinate within the current panel of the number widget in pixels

e.g. :y 200

The **:x** and **:y** parameters may be omitted in which case the packer geometry manager will be used and the position of the window will be determined by the contents of the master (panel) window. If however the **:x** and **:y** parameters are supplied then the placer geometry manager will be used to position the window within the master (panel) window.
Warning: Note that it is not possible to mix use of the packer and placer geometry managers within the same window. Thus all windows within a panel must request or imply that the same geometry manager is used.

:w *number*

number is the width of the entry window in characters.
e.g. :w 20

:h *number*

number is the height of the number widget in characters.
e.g. :h 2
The default value is :h 1.

:value *value*

value is the initial value of the number.
The default value is 0.

:min *value*

value is the minimum value of the number.

:max *value*

value is the maximum value of the number.

:**inc** *value*

value is the increment (or decrement) made to the value of the number by the up (or down) buttons.

:valuex *value*

value is the position of the left hand edge of the entry window and the right hand edge of the label window with relation to the left hand edge of the checkbox widget. It is thus the space available to write the label string which is right justified within the label window and also allows alignment of the entry windows of several widgets. The value may be expressed in units of pixels or characters.
e.g. :valuex 100 (pixels); :valuex 15c (characters)
By default the component windows of the widget are packed within the widget frame..

:label *string*

string is a text string to be written into the label component of the number widget.
The default is the name of the number widget.

See also common procedure arguments defined later in this document.

procedure

:gauge name

Creates a gauge widget within the current panel frame for the TCL interpreter. A gauge widget is a frame window containing three label windows and a scale window.
name is used to generate the path name of the windows to be created. It should have the format of a widget name (e.g. :gauge GG) and should be unique within this interpreter. The name will be lower-cased (using the *string tolower* command) in order to generate the path name used to create the windows.
A gauge widget is not interactive.
The path names generated have the root *framename.panelname.gauge-name* and the items *label, min, max* and *scale.*
The **:gauge** procedure may also be used directly within a frame window (i.e. before any panel window has been created).

The **:gauge** procedure may have a number of arguments however none of these is mandatory.

arguments

:x *number*

number is the x-co-ordinate within the current panel of the gauge widget in pixels

e.g. :x 400

:y *number*

number is the y-co-ordinate within the current panel of the gauge widget in pixels

e.g. :y 200

The **:x** and **:y** parameters may be omitted in which case the packer geometry manager will be used and the position of the window will be determined by the contents of the master (panel) window. If however the **:x** and **:y** parameters are supplied then the placer geometry manager will be used to position the window within the master (panel) window.
Warning: Note that it is not possible to mix use of the packer and placer geometry managers within the same window. Thus all windows within a panel must request or imply that the same geometry manager is used.

:w *number*

number is the width of the gauge widget in pixels.

e.g. :w 200

The default action is to allow the geometry manager to determine the size of the widget.

:h *number*

number is the height of the gauge widget in pixels.

e.g. :h 24

The default value is 1 character height.

:value *value*

value is the initial value of the gauge scale and determines the position of the divider.
The default value is 0.

:min *value*
value is the minimum value of the gauge scale.

:max *value*
value is the maximum value of the gauge scale.

:valuex *value*
value is the position of the right hand edge of the label window with relation to the left hand edge of the scale widget. It is thus the space available to write the label string which is right justified within the label window. The value may be expressed in units of pixels or characters.
e.g. :valuex 100 (pixels); :valuex 15c (characters)

:label *string*
string is a text string to be written into the label component of the gauge widget.
The default is the name of the gauge widget.

:gauge-width *value*
value is the width of the gauge widget in pixels.
The default value is 80.

:gauge-height *value*
value is the length of the gauge widget in pixels.
The default value is 1/10 of the gauge width.

:max-tick-label *string*
string is a text string to be used for the label at the right hand end of the gauge.
By default the label shows the maximum value of the gauge scale.

:min-tick-label *string*
string is a text string to be used for the label at the left hand end of the gauge.
By default the label shows the minimum value of the gauge scale.

See also common procedure arguments defined later in this document.

procedure

:slider *name*

Creates a slider widget within the current panel frame for the TCL interpreter. A slider widget is a frame window containing three label windows, an entry window and a scale window.

name is used to generate the path name of the windows to be created. It should have the format of a widget name (e.g. :slider GG) and should be unique within this interpreter. The name will be lower-cased (using the *string tolower* command) in order to generate the path name used to create the windows.

When an event is generated either by pressing the Return key after entering a value in the entry window or by releasing mouse button 1 while the mouse focus is on the slider window then ,if available, the TCL script supplied using the **:command** parameter will be executed. If this is not available the TCL command

click FRAME “” “” SLIDER value

will be executed where **FRAME** and **SLIDER** are widget names and **value** is the current value of the slider widget scale.

The path names generated have the root *framename.panelname.slidername* and the items *label, entry, min, max* and *scale.*

The **:slider** procedure may also be used directly within a frame window (i.e. before any panel window has been created).

Normally it is necessary when typing information into the entry window to press the *return* key in order to generate an event and cause the procedure linked to the entry to be executed. Often users fail to do this and so a binding is made to the mouse focus and the contents of the entry window are checked whenever the mouse focus leaves the entry window. If information is found to have been typed into the entry window which has not been seen by a *return* key event then an action is taken as is the *return* key had been pressed.

The **:slider** procedure may have a number of arguments however none of these is mandatory.

arguments

:x *number*

number is the x-co-ordinate within the current panel of the slider widget in pixels

e.g. :x 400

:y *number*

number is the y-co-ordinate within the current panel of the slider widget in pixels

e.g. :y 200

The **:x** and **:y** parameters may be omitted in which case the packer geometry manager will be used and the position of the window will be determined by the contents of the master (panel) window. If however the **:x** and **:y** parameters are supplied then the placer geometry manager will be used to position the window within the master (panel) window.

Warning: Note that it is not possible to mix use of the packer and placer geometry managers within the same window. Thus all windows within a panel must request or imply that the same geometry manager is used.

:w *number*

number is the width of the slider widget in pixels.

e.g. :w 200

The default action is to allow the geometry manager to determine the size of the widget.

:h *number*

number is the height of the slider widget in pixels.

e.g. :h 24

The default value is 1 character height.

:value *value*

value is the initial value of the slider widget scale and determines the position of the divider.

The default value is 0.

:min *value*

value is the minimum value of the slider widget scale.

:max *value*

value is the maximum value of the slider widget scale.

:valuex *value*

value is the position of the right hand edge of the label window with relation to the left hand edge of the slider widget. It is thus the space available to write the label string which is right justified within the label window. The value may be expressed in units of pixels of characters.

e.g. :valuex 100 (pixels); :valuex 15c (characters)

:label *string*

string is a text string to be written into the label component of the slider widget.

The default is the name of the slider widget.

:ticks **T|F**

If the option **T** is selected then numerical tick marks are displayed below or to the left of the slider.

The default is **F** - no tick marks are displayed.

:layout **H|V**

If the option **V** is selected then the slider widget is displayed in a vertical orientation otherwise it is displayed in a horizontal orientation.

The default is that the widget is displayed horizontally.

:scale-length *value*

value is the desired long dimension of the scale in pixels.

The default value is 100.

:scale-width *value*

value is the desired narrow dimension of the scale in pixels.

The default value is 15.

:slider-width *value*

value is the desired size of the slider in pixels.

The default value is 10.

:limit-width *value*

value is the desired width of the labels (in characters) drawn at the ends of the slider scale.

The default is that the label widths are determined by the text strings written in them.

:entry-width *value*

value is the desired width of the entry widget in pixels.

The default width is 6 pixels.

:resolution *value*

value is the desired resolution of the scale window in units per step.

By default the resolution is calculated by {absolute value of (max - min)) / (steps)

:steps *value*

value is the desired number of steps in the scale window.

The default value of step is 256. This parameter is ignored unless **:resolution** is also supplied.

:command *string*

string is any TCL command or commands to be executed when an event is generated.

If a command is supplied and executed in this manner then the **click** procedure is not invoked

See also common procedure arguments defined later in this document.

procedure

:menubutton *name*

Creates a menubutton widget within the current panel frame for the TCL interpreter. A menubutton widget is a frame window containing a menubutton window and menu.
name is used to generate the path name of the windows to be created. It should have the format of a widget name (e.g. :menubutton MENU) and should be unique within this interpreter. The name will be lower-cased (using the *string tolower* command) in order to generate the path name used to create the windows.
When a menu item is selected by releasing mouse button 1 while the mouse focus is on a menu item then ,if available, the TCL script supplied using the **:command** parameter will be executed. If this is not available the TCL command

click FRAME “ ” MENUBUTTON item index

will be executed where **FRAME** and **MENUBUTTON** are widget names and **item** is the contents of the menu entry selected. If mouse button 3 is released while the mouse focus is on the menubutton window then the default menu entry is returned as **item**. **index** is the index (from zero) of the selected item in the menu.
The path names generated have the root *framename.panelname.menu-buttonname* and the item menu*.*
The **:menubutton** procedure may also be used directly within a frame window (i.e. before any panel window has been created).

The **:menubutton** procedure may have a number of arguments however none of these is mandatory.

arguments

:x *number*

number is the x-co-ordinate within the current panel of the menubutton widget in pixels

e.g. :x 400

:y *number*

number is the y-co-ordinate within the current panel of the menubutton widget in pixels

e.g. :y 200

The **:x** and **:y** parameters may be omitted in which case the packer geometry manager will be used and the position of the window will be determined by the contents of the master (panel) window. If however the **:x** and **:y** parameters are supplied then the placer geometry manager will be used to position the window within the master (panel) window.
Warning: Note that it is not possible to mix use of the packer and placer geometry managers within the same window. Thus all windows within a panel must request or imply that the same geometry manager is used.

:w *number*

number is the width of the menubutton window in characters.

e.g. :w 20

:h *number*

number is the height of the menubutton window in lines of text.

e.g. :h 2

The **:w** and/or **:h** parameters may be omitted in which case the size of the menubutton will be determined by its contents. If however the **:w** and **:h** parameters are supplied then these will be used to **configure** the button window. Normally **:h** will not be supplied.

:value *value*

value is the index of the default menu item within the menu.

The default value is 0.

:label *string*

string is a text string to be written onto the menubutton window.

The default is the name of the menubutton widget.

:strings *list*

list is a simple TCL list of text strings which form the elements of the menu.

The default is a null list and hence an empty menu.

:menu *list*

list is a TCL list of items which form the elements of the menu. Each item may be a simple string in which case it is appended as the name of a command in the current menu. However an item may itself be a list in which case the first element of the item is appended as the name of a cascade in the current menu and the remaining elements of the item are used to construct the cascade menu.

It is not meaningful that **:menu** and **:string** are both specified in the same command.

:command *string*

string is any TCL command or commands to be executed when an event is generated.

If a command is supplied and executed in this manner then the **click** procedure is not invoked

The arguments **:rows** and **:cols** are no longer available since the use of **:menu** permits menus to be structured as required.

See also common procedure arguments defined later in this document.

procedure

:optionbutton *name*

Creates an optionbutton widget within the current panel frame for the TCL interpreter. An optionbutton widget is a button window. A list of options is associated with the widget and the currently selected option is displayed within the button window. Items from the option list are selected cyclically using the mouse.
name is used to generate the path name of the window to be created. It should have the format of a widget name (e.g. :optionbutton OPT) and should be unique within this interpreter. The name will be lower-cased (using the *string tolower* command) in order to generate the path name used to create the window.
When the button is selected using the mouse then ,if available, the TCL script supplied using the **:command** parameter will be executed. If this is not available the TCL command

click FRAME " " OPTIONBUTTON index option

will be executed where **FRAME** and **OPTIONBUTTON** are widget names, **index** is the index (from zero) into the option list of the current option and **option** is the current option.
The path name generated has the form *framename.panelname.buttonname.*
The **:optionbutton** procedure may also be used directly within a frame window (i.e. before any panel window has been created).

The **:optionbutton** procedure may have a number of arguments however none of these is mandatory.

arguments

:x *number*

number is the x-co-ordinate within the current panel of the optionbutton widget in pixels

e.g. :x 400

:y *number*

number is the y-co-ordinate within the current panel of the optionbutton widget in pixels

e.g. :y 200

The **:x** and **:y** parameters may be omitted in which case the packer geometry manager will be used and the position of the window will be determined by the contents of the master (panel) window. If however the **:x** and **:y** parameters are supplied then the placer geometry manager will be used to position the window within the master (panel) window.
Warning: Note that it is not possible to mix use of the packer and placer geometry managers within the same window. Thus all windows within a panel must request or imply that the same geometry manager is used.

:w *number*

number is the width of the optionbutton window in characters.

e.g. :w 20

:h *number*

number is the height of the optionbutton window in lines of text.

e.g. :h 2

The **:w** and/or **:h** parameters may be omitted in which case the size of the optionbutton will be determined by its contents. If however the **:w** and **:h** parameters are supplied then these will be used to **configure** the button window. Normally **:h** will not be supplied.

:value *value*

value is the index of the default option item within the option list.

The default value is 0.

:strings *list*

list is a simple TCL list of text strings which form the elements of the option list.

The default is a null list and hence an empty option list. **:menu** is an alias for **:strings**.

:command *string*

string is any TCL command or commands to be executed when an event is generated.

If a command is supplied and executed in this manner then the **click** procedure is not invoked

See also common procedure arguments defined later in this document.

procedure

:choice *name*

Creates a choice widget within the current panel frame for the TCL interpreter. A choice widget is a frame window containing a number of checkbutton windows and (optionally) a label window.
name is used to generate the path name of the windows to be created. It should have the format of a widget name (e.g. :choice CH) and should be unique within this interpreter. The name will be lower-cased (using the *string tolower* command) in order to generate the path name used to create the windows.
When a choice is selected by releasing mouse button 1 while the mouse focus is on the choice then the TCL command

click FRAME " " CHOICE value

will be executed where **FRAME** and **CHOICE** are widget names and **value** is determined by which choice or choices are currently selected (see :exclusive).
The path names generated have the root *framename.panelname.choice-name* and the items *label* and *button*.*
The **:choice** procedure may also be used directly within a frame window (i.e. before any panel window has been created).

The **:choice** procedure may have a number of arguments however none of these is mandatory.

arguments

:x *number*

number is the x-co-ordinate within the current panel of the choice widget in pixels

e.g. :x 400

:y *number*

number is the y-co-ordinate within the current panel of the choice widget in pixels

e.g. :y 200

The **:x** and **:y** parameters may be omitted in which case the packer geometry manager will be used and the position of the window will be determined by the contents of the master (panel) window. If however the **:x** and **:y** parameters are supplied then the placer geometry manager will be used to position the window within the master (panel) window.
Warning: Note that it is not possible to mix use of the packer and placer geometry managers within the same window. Thus all windows within a panel must request or imply that the same geometry manager is used.

:w *value*

value is the width of the checkbutton windows in characters.

e.g. :w 10

:value *value*

value determines the initially selected checkbutton or checkbuttons (see :exclusive).

The default value is 0 (no items selected).

:valuex *value*

value is the position of the right hand edge of the label window with relation to the left hand edge of the choice widget. It is thus the space available to write the label string but also allows alignment of the checkbutton windows of several choice widgets. The value may be expressed in units of pixels or characters.

e.g. :valuex 100 (pixels); :valuex 15c (characters)

The default value is 0. If :valuex is specified then a label window will be created even if :label is not specified (in which case the label will be a single space character).

:label *string*

string is a text string to be written into the label component of the choice widget.

By default no label window is created. A non null :label string (a single space is adequate) causes the label window to be created (see also :valuex)..

:strings *list*

list is a TCL list of strings. For each item in the list a checkbutton window is created and the item string is written onto the checkbutton.

The default is a null list.

:layout **V|H**

defines the layout (**V**ertical or **H**orizontal) of the components of the choice widget.

The default layout is H.

:exclusive **T|F**

If **T**rue then only one checkbutton at a time may be selected and :value defines the index of the selected checkbutton within the list of available checkbuttons. If **F**alse then any number of checkbuttons may be selected and value is a bitmap which defines the selected checkbuttons within the list of available checkbuttons. The low order bit of the bitmap corresponds to checkbutton index = 0.

The default is T.

See also common procedure arguments defined later in this document.

procedure

:choice-stack *name*

Creates a choicestack widget within the current panel frame for the TCL interpreter. A choicestack widget is a frame window containing a menubutton window, menu, message window and (optionally) label window.
name is used to generate the path name of the windows to be created. It should have the format of a widget name (e.g. :choicestack CHSTK) and should be unique within this interpreter. The name will be lower-cased (using the *string tolower* command) in order to generate the path name used to create the windows.
When a choice is selected by releasing mouse button 1 while the mouse focus is on an item in the menu then the TCL command

click FRAME " " CHOICESTACK index item

will be executed where **FRAME** and **CHOICESTACK** are widget names, **index** is the index of the selected menu item within the choicestack menu and **item** is the contents of the menu item selected.
The selected menu item is written into the message window.
The path names generated have the root *framename.panelname.choice-stackname* and the items *menu, message* and *label.*
The **:choice-stack** procedure may also be used directly within a frame window (i.e. before any panel window has been created).

The **:choice-stack** procedure may have a number of arguments however none of these is mandatory.

arguments

:x *number*

number is the x-co-ordinate within the current panel of the choicestack widget in pixels

e.g. :x 400

:y *number*

number is the y-co-ordinate within the current panel of the choicestack widget in pixels

e.g. :y 200

The **:x** and **:y** parameters may be omitted in which case the packer geometry manager will be used and the position of the window will be determined by the contents of the master (panel) window. If however the **:x** and **:y** parameters are supplied then the placer geometry manager will be used to position the window within the master (panel) window.
Warning: Note that it is not possible to mix use of the packer and placer geometry managers within the same window. Thus all windows within a panel must request or imply that the same geometry manager is used.

:value *value*

value is the index of the initially selected item within the choice menu. The selected menu item is written into the message window.
The default value is 0.

:valuex *value*

value is the position of the left hand edge of the menubutton window with relation to the left hand edge of the checkbox widget. It is thus the space available to write the label string but also allows alignment of the menubutton windows of several checkbox widgets. The value may be expressed in units of pixels or characters.

e.g. :valuex 100 (pixels); :valuex 15c (characters)

The default value is 0. If :valuex is specified then a label window will be created even if :label is not specified (in which case the label will be a single space character).

:label *string*

string is a text string to be written into the label component of the choice-stack widget.

By default no label window is created. A non null :label string (a single space is adequate) causes the label window to be created (see also :valuex).

:strings *list*

list is a simple TCL list of text strings which form the elements of the menu.

The default is a null list and hence an empty menu.

:menu *list*

list is a TCL list of items which form the elements of the menu. Each item may be a simple string in which case it is appended as the name of a command in the current menu. However an item may itself be a list in which case the first element of the item is appended as the name of a cascade in the current menu and the remaining elements of the item are used to construct the cascade menu.

It is not meaningful that **:menu** and **:string** are both specified in the same command.

The arguments **:rows** and **:cols** are no longer available since the use of **:menu** permits menus to be structured as required.

See also common procedure arguments defined later in this document.

procedure

:spectrum *name class action args*

Creates a spectrum widget within the current panel frame for the TCL interpreter.
name is used to generate the path name of the window to be created. It should have the format of a widget name (e.g. :spectrum s0) and should be unique within this interpreter. The name will be lower-cased (using the *string tolower* command) in order to generate the path name used to create the window.

The **:spectrum** procedure has a number of arguments which determine the action required.

arguments

:class *name*

name is the class of action to be performed.

:action *name*

name is the action required which is a member of the current class.

:args *list*

list are arguments required by the current action. Not all actions require an args list.

spectrum

create	:x *value* :y *value* :w *value* :h *value*
overlap	[name]
removeoverlaps	
add	[name]
remove	
squeeze	
current	[|m|ptr]
title	[|n]
size	
name	[|m]
save	[n]
get	
overlap-dataset	
data	
linestyle	
refresh	
info	
add-dataset	
clear	

get
- overlay-dataset [i j]
- data [base range counts]
- linestyle [0|1|2]
- refresh
- info
- add-dataset [i j]
- size [i j] | [i j k l]

refresh
- tags
- calibrations

configure
- -colourmap
- -overlapcolourmap
- -style
- -select

cget

view
- shift [n]
- scale [n]
- reset
- undo
- redo
- get
- current

get
- dimension
- channel [x y]
- count [x y]
- calibration [x y]
- channelx [x y]
- channely [x y]
- calibrationx [x y]
- calibrationy [x y]
- decalibrationx [l]
- decalibrationy [l]
- box

gate		
	pop	[m]
	push	[l m]
	get	[m]
	set	[x1,y1 x2,y2]
	vertex hit	[x y]
	vertex add	
	vertex remove	
	vertex set	
	get2d	
	select	
	set2d	
pointer		
	pop	
	get	
	push	
	set	
	size	
tag		
	near	x y
	set	x y
	get	n
	push	n
	pop	n
	get2d	n
	push2d	x y
	move	

cut   x | y   chan   matrixptr   cutptr

procedure

:canvas *name*

Creates a canvas widget within the current panel frame for the TCL interpreter.
name is used to generate the path name of the window to be created. It should have the format of a widget name (e.g. :canvas CVS) and should be unique within this interpreter. The name will be lower-cased (using the *string tolower* command) in order to generate the path name used to create the window.
Currently the canvas widget is non interactive and thus no **click** procedure is invoked.
The **:canvas** procedure may also be used directly within a frame window (i.e. before any panel window has been created).

The **:canvas** procedure may have a number of arguments however none of these is mandatory.

arguments

:x *number*

number is the x-co-ordinate within the current panel of the canvas widget in pixels

e.g. :x 400

:y *number*

number is the y-co-ordinate within the current panel of the canvas widget in pixels

e.g. :y 200

The **:x** and **:y** parameters may be omitted in which case the packer geometry manager will be used and the position of the window will be determined by the contents of the master (panel) window. If however the **:x** and **:y** parameters are supplied then the placer geometry manager will be used to position the window within the master (panel) window.
Warning: Note that it is not possible to mix use of the packer and placer geometry managers within the same window. Thus all windows within a panel must request or imply that the same geometry manager is used.

:w *number*

number is the width of the canvas widget in pixels

e.g. :w 400

:h *number*

number is the height of the canvas widget in pixels

e.g. :y 200

The **:canvas** procedure creates an empty window into which objects can be created using the **:draw** procedure.

procedure

:draw *object*

Creates an item of type *object* on the current canvas. *object* should be one of the following: arc, bitmap, image, line, oval, polygon, rectangle, text,window.

The **:draw** procedure may have a number of arguments however none of these is mandatory.

arguments

:colour *string*

string is the colour to be used when drawing the current object. It is used to configure the object using the **-fill** option.

e.g. :colour red

:text *string*

string is the text to be used when drawing the current object. It is used to configure the object using the **-text** option.

e.g. :text “hello word”

:tag *name*

name is used to configure the object using the **-tags** option

:arrow *string*

string is used to configure the object using the **-arrow** option. *string* should have one of the values; none, first,last or both.

:delete **T**

The tag specified by a previous **:tag** argument is to be deleted.

e.g. :draw text :tag l0 :delete T

:value *list*

list are items to be used when creating the object. All arguments remaining in the procedure call are used in this way and so **:value** must be the last argument.

e.g. :draw line :value 20 0 20 10

This argument provides a way by which any selection of options can be supplied to configure the object.

e.g. :draw arc :value 10 10 30 40 -extent 45 -outline blue

Common Procedure Arguments

procedure

All except :frame and :panel

The following arguments are available for all procedures (except **:frame** and **:panel**).

:show **T|F**

If the value is **T**rue then the window or widget is made visible; if **F**alse then the window or widget is made non visible.
The default is that widgets are visible.

:deaf **T|F**

If the value is **T**rue then the window or widget is made non-interactive; if **F**alse then the window or widget is made interactive.
The default is that widgets are interactive. Note however that some widgets (e.g. panel and message) are only non-interactive.

:inactive **T|F**

If the value is **T**rue then the window or widget is made inactive (displayed dimmer than normal); if **F**alse then the window or widget is made active (normal state). The window or widget will also be made non-interactive.
The default is that widgets are active.

:bold **T|F**

If the value is **T**rue then text displayed within the window or widget will use a bold font; if **F**alse then text will be displayed using a normal font.
The default is that a normal font is used.

:**double-click** **T|F**

If the value is **T**rue then interactive windows and widgets are bound to a double mouse click event in addition to the normal single mouse click event. The first click causes the **click** procedure to be entered and the second click causes the **double-click** procedure to be entered. If the value is **F**alse then only single mouse click events are enabled.
The default is that only single click events are enabled.

:help ***text***

text is information to be used for level 1 (task local) help. For further information see the section on the **Help** System later in this document.

:helpitem ***name***

name is the name to be used when storing and accessing level 1 help. . For further information see the section on the **Help** System later in this document.
The default is that the widget name is used.

All except :frame

The following arguments are available for all procedures (including **:panel**).

:after *widget*

widget defines a widget name to be used by the packer geometry manager when inserting the widget into the pack list. The current widget is inserted into the list after *widget*. This also becomes the default action for all further widgets within the current frame widget.

:before *widget*

widget defines a widget name to be used by the packer geometry manager when inserting the widget into the pack list. The current widget is inserted into the list before *widget*. This also becomes the default action for all further widgets within the current frame widget.

Additional procedures

The following procedures are available at any time after the first **:frame** procedure has been called

procedure

:path *widget*

Given a widget name the full TK path name of the window created (or of the frame window created to hold a complex widget) is returned.

e.g. :path MSG may return .app.panel.msg

procedure

:show T|F

If the value is **T**rue then the master window is made visible; if **F**alse then the master window is made non visible.

It is normally necessary for a :**show T** procedure call to be made at the end of definition of the master window.

The **:show** procedure also configures the footer width to the width of the frame and sets the keyboard focus to follow the mouse position. The procedure **open-frame** also causes the window to be made visible but does not define the keyboard focus. This is useful if an application for some reason requires a different keyboard focus algorithm.

procedure

:set *item* *value*

Allows access to the internal constants used by the procedures. The constant held in the TCL global variable _*item* is set to *value*.

Current constants and their default values are :

listch (17); listcw (7); chkbtsize (24); chcesize (24); charh (25); charw (7); footercharw (6)

The **:set** procedure can be used to redefine options to be used by the packer geometry manager for future widgets.

layout	default	left
anchor	default	nw
expand	default	yes
fill	default	both
ipadx	default	0
ipady	default	0
padx	default	0
pady	default	0
footer-position	default	left

procedure

:enable *widget* *events*

Allows additional events to be bound to a window. *widget* is either a widget name (it does not start with a period) or is a window name (it does start with a period).*events* are a list of TK events which are to be bound to the widget/window.

e.g. **:enable F1 Enter**

This causes an event to be generated when the mouse enters the window identified by F1. The TCL procedure

click frame :Enter widget

will be invoked.

procedure

:help *widget* *text*

text defines information to be used for level 1 (task local) help for *widget.* For further information see the section on the **Help System** later in this document.

procedure

:colour *mode window* [*option value*]

Defines the colour to be used for future widgets.

mode can be either **local** (the colour selection applies only to this interpreter) or **global** (the colour selection will apply to the whole session).

window is the window type and is one of: frame, label, entry, listbox, text, message, scale, scrollbar, button, checkbutton, menubutton, canvas.

option is the window component and is one of: activebackground, activeforeground, background, disabledforeground, highlightbackground, highlightcolor, insertbackground, selectcolor, selectbackground, selectforeground, troughcolor.

value is the colour to use for the selected item.

Note - not all options are valid for all window types.

procedure

:font *mode window weight* [*component value*]

Defines the font to be used for future widgets.

window is the window type and is one of: label, entry, listbox, text, message, scale, button, checkbutton, menubutton, entry, menu.

mode can be either **local** (the font selection applies only to this interpreter) or **global** (the font selection will apply to the whole session).

weight is the font weight and is either **b** (bold) or **n** (medium)

component is the window component and is one of: :fndry, :fmly, :wght, :pxlsz, :ptsz, :slant, :swdth, :rgstry.

value is the value to use for the selected component.

Default values: fndry=adobe; fmly=helvetica; pxlsz=12; ptsz=120; slant=r; swdth=normal; rgstry=iso8859.

The following procedures are useful following a **:panel** procedure call to modify options used in the creation of widgets within the panel;.

procedure

:layout L|R|T|B|H|V

The value supplied determines which side of the master should be used when packing slave windows into the master. Left, Right,Top and Bottom are layout options specific to the packer geometery manager and the geometry manager flag is set so that the packer is used for subsequent widgets within the current panel. Horizontal and Vertical are options which select the placer geometry manager and the geometry manager flag is set so that the placer is used for subsequent widgets within the current panel.

arguments

:pack **T|F**

If the value is True then the value of the geometry manager flag is set so that the packer is used by default for subsequent widgets within the current panel; if the value is False then the value of the geometry manager flag is set so that the placer is used by default for subsequent widgets within the current panel.

:place **T|F**

If the value is True then the value of the geometry manager flag is set so that the placer is used by default for subsequent widgets within the current panel; if the value is False then the value of the geometry manager flag is set so that the packer is used by default for subsequent widgets within the current panel.

:anchor *where*

where should be a valid window point to be used when placing or packing slave windows within their master window. The parameter supplied is used by default for subsequent widgets within the current panel.
Valid points are n,s,e,w,nw,ne,se,sw.

procedure

:row-gap *value*

See the **:row-gap** argument of the **:panel** procedure. This procedure allows the current row gap to be changed. It also causes the geometry manager selected to be the placer.

procedure

:col-gap *value*

See the **:col-gap** argument of the **:panel** procedure. This procedure allows the current column gap to be changed. It also causes the geometry manager selected to be the placer.

procedure

:next-row **T**

Sets the current x position (used when placing widgets) to 0 and increments the y position (used when placing widgets) by the height of the last widget created plus any additional space specified by the row-gap parameter. It also causes the geometry manager selected to be the placer.

procedure

:next-col **T**

Sets the current y position (used when placing widgets) to 0 and increments the x position (used when placing widgets) by the width of the last widget created plus any additional space specified by the col-gap parameter. It also causes the geometry manager selected to be the placer.

procedure

:hskip *number*

number is a value in pixels to be added to the current x position (used when placing widgets). *number* may be positive in which case positioning skips to the right or it may be negative in which case positioning skips to the left.

procedure

:vskip *number*

number is a value in pixels to be added to the current y position (used when placing widgets). *number* may be positive in which case positioning skips downwards or it may be negative in which case positioning skips upwards.

Changing existing widget options

Once created many of the options specified for the **widget** may subsequently be changed. This is particularly useful for adding to or changing the contents of **list** and **galley** widgets. However menus can be dynamically defined by changing the list of strings which is used to specify the menu list.

procedure

:modify *name* *arguments*

arguments define changes required to the widget known by ***name.***
The changes possible are determined by the type of widget. Unless specified otherwise the definition of the procedure which created the widget should be referenced for a definition of the arguments.

Widget type **frame**
arguments

:label	*string*	
:w	*number*	
:h	*number*	
:closed	**T|F**	
:delete	**T**	deletes the frame widget
:busy	**T|F**	see footer description
:transient	**T**	

Widget type **panel**
arguments

:w	*number*	
:h	*number*	
:delete	**T**	deletes the panel widget

Widget type **button**
arguments

:deaf	**T|F**	
:inactive	**T|F**	
:busy	**T|F**	if **T** then button show sunken
:label	*string*	
:colour	*value*	

Widget type **text**
arguments

:label	*string*
:w	*number*
:value	*string*
:deaf	**T|F**
:inactive	**T|F**

Widget type **list**
arguments

:strings	*list*
:rows	*number*
:cols	*number*
:show	**T|F**
:see	*number*
:select	*value*
:deselect	*value*
:empty	**T|F**
:append	**T|F**

Widget type **galley**
arguments

:strings	*list*
:rows	*number*
:cols	*number*
:see	*number*
:show	**T|F**
:inactive	**T|F**
:empty	**T|F**
:append	**T|F**
:usingtag	*number*
:createtag	*number*
:deletetag	*number*

:usingtag defines the tag to be used by subsequent **:strings** arguments.
:createtag creates (configures) a tag in the text widget and **:deletetag** delete a tag.

Widget type	**message**	
arguments		
	:label	*string*
	:font	*string*
	:colour	*string*
	:bold	**T|F**

Widget type	**checkbox**	
arguments		
	:value	*value*
	:deaf	*T|F*
	:inactive	*T|F*
	:colours	*list*

Widget type	**number**	
arguments		
	:deaf	*T|F*
	:inactive	*T|F*
	:label	*string*
	:value	*value*

Widget type	**gauge**	
arguments		
	:label	*string*
	:value	*value*
	:min	*value*
	:max	*value*
	:max-tick-value	*value*
	:inactive	**T|F**

Widget type	**slider**	
arguments		
	:label	*string*
	:value	*value*
	:min	*value*
	:max	*value*
	:inactive	**T|F**

Widget type	**menubutton**	
arguments		
	:strings	*list*
	:menu	*list*
	:label	*string*
	:inactive	**T|F**

Widget type	**optionbutton**	
arguments		
	:strings/:menu	*list*
	:value	*value*
	:inactive	**T|F**

Widget type	**choice**	
arguments		
	:value	*value*
	:inactive	**T|F**

Widget type	**choice-stack**	
arguments		
	:strings	*list*
	:menu	*list*
	:value	*value*
	:inactive	**T|F**

Widget type	**canvas**	
arguments		
	:w	*number*
	:h	*number*
	delete	**T|F**

The Frame Footer and Busy Indicator

The argument **:show-footer T** of the frame procedure causes the footer to be created for the frame widget. The footer is a frame window having a width equal to the width of the frame widget and a height of 1 character. It is positioned at the bottom of the frame widget.
The footer widget contains a message window which is used to hold short term information messages from the application. By default the message is left justified by this can be modified by the command

:set footer-position location

executed **before** the :frame procedure call which creates the footer.
The path names generated have the root *framename.footer* and the item *footer.* The widget name FOOTER is generated allowing reference to the footer widget in the same way as all other widgets.
Messages are inserted into the footer by the procedure

set-footer [frame] string

where **frame** is the name of a FRAME widget which may be omitted in which case the main frame for the application is used and **string** is the text to be inserted into the footer. The text will if necessary be truncated in order to fit the single line of the footer.
Messages are removed by the procedure **clear-footer [frame]** which is equivalent to **set-footer [frame] ""**.
A binding is made such that clicking mouse button 1 on a footer message causes the message to be cleared.

The busy indicator is normally used by the application to indicate that it is performing some action which may take more than the normal interactive response time. Typically this is used during networked server accesses and when spanning a shell to execute a program in foreground. The indicator is created by configuring the background colour of the footer widget windows. The colour used is taken from the task variable **tkfooterbusycolor**. It has the default value **Pink**.
If the footer message is empty at the time that the busy indicator is enabled then the message **Busy....** is additionally put into the footer.
The busy indicator is controlled by the command

:modify FRAME **:busy T|F**

The procedures **set-busy** and **clear-busy** are also provided.
A binding is made such that clicking mouse button 3 on the footer area causes the busy indicator to be cleared.

Window Auto-Size

If the packer window manager is used throughout then the sizes of panels and frames is determined automatically by their contents. However, if any option is used which requires that the placer window manager is used (for example the use of rows and columns for laying out panels) then it is necessary that the panel is given a explicit size.
The library routines when placing widgets in panels track the maximum x and y co-ordinates used and this information can be used to supply the width and height needed for the panel. For this feature to be enabled the **:panel** command must not supply a width (**:w**) or height (**:h**) when creating the panel. It no explicit size is given to a panel then the autosize feature will supply it. This feature can also be disabled by use of the command **:set autosize 1**.
Autosize is only available for the **panel** widget. It is expected that normally no absolute positioning information (**:x** and **:y**) will be given via the **:panel** command and that panel widgets are then packed into the frame widget which will then be autosized by the geometry manager.
Note that the **:show T** command at the end of the frame creation will configure the frame footer (if it exists) to be the width of the frame.

Text windows created by the **:galley** procedure do not usually resize automatically to best fill the window when the frame is resized to increase its size. (The window manager does seem to respond correctly when the frame is made smaller).
The **:galley_size** procedure should be called at the end of the creation of a frame containing a galley and after the frame has been opened in order to establish initial sizes of the windows.
The galley would have been created with something like

:galley $W :rows $rows :cols $cols

and the :galley_size procedure would then be

:galley_size $W $rows $cols

The **:galley_resize** procedure should be called whenever it is required to check if the galley can be resized to fit the window.

:galley_resize $W

Typically a binding is made to create an event whenever the mouse moves into or out of the frame.

:enter [self frame-name] Enter

Help System

The Help System for the **Midas** package splits into 2 components -

1) features which are executed within each task

These facilities give immediate help on the use of each **widget** within the task using the window footer and slightly more information using a pop-up dialog window.

2) features which are executed within the **help** task.

These facilities allow for more extensive help possibly envoking an external viewer program to show a document. The response time may be measured in several seconds if an external program has to be started.

Task local features

These require that help information is available in the **TCL** array **:help**. Help is linked to a **widget** and (by default) help information for **widget FRED** is in **:help(FRED).**
It is possible to specify the help item name to be used by the parameter **:helpitem *name*** in the widget creation procedure call. For example

:button ATTACH :helpitem CONNECT

would create a button labelled as **Attach** but take help information from **:help(CONNECT)**.

If the array element **DEFAULTHELP** exists then this information is used for any widget which does not have its own private help information.

The parameter **:help F** in the widget creation procedure call causes setup of the bindings which implement the task local help for a widget to be bypassed.

The task **TCL** script can load help information in one of 4 ways

1) directly by a set command -

set :help(FRED) "This is the FRED widget"

2) using the **:help** procedure -

:help FRED "This is the FRED widget"

3) as an argument to the procedure creating the widget -

:button FRED :help "This is the FRED widget"

4) within the file **$taskname.txt** which is searched for using the **TCLPATH** variable. Normally this would imply that for a task there are 2 files **$taksname.tcl** and **$taskname.txt**

The format of the **.txt** file is

+widget FRED
This is the FRED widget

The FRED widget is a really wonderful thing
+widget JIM

etc

The first line of the help information for the **widge**t is put into the window footer while the mouse is over the **widget** and removed when the mouse leaves the window. Since this action can in some cases cause problems by overwritting other useful footer information it can be controlled by the **TCL** global variable **_suppresshelp**. Setting **_suppresshelp** to 0 enables the action and setting **_suppresshelp** to 1 disables it.

The whole of the help information is shown in a dialog window if button **F1** is pressed while the mouse is over the **widget**.

To implement the level 1 help feature bindings are made by all **widget** creation procedures for the events **Enter** and **Leave** (unless suppressed by the **:help F** parameter).

Facilities within the help task

1) These can be called by creating a button using
:button HELP :command click-help

The default **click-help** procedure then calls the help task **help-on-task** procedure with (by default) the calling task's name.
The task name used in the call to the Help task can be specified by the **TCL** variable ***helpname***.
For example: **click-help** in the task **ap** will invoke **help-on-task ap** but if **set helpname apparatus** were specified then **click-help** would invoke **help-on-task apparatus**.

2) Pressing the **Shift-F1** button while the mouse is over a window calls (by default) the **help** task **help-on-widget** procedure with the **widget** name.

Note that when the mouse is positioned over the window background the **help-on-task** procedure is called.
The task name and widget name to be used can be specified as described previously. For example in the task **ap** widget **ATTACH** would invoke **help-on-widget ap ATTACH** but this could be changed as shown to **help-on-widget apparatus CONNECT.**

The **help** information will be searched for taking the **TCLPATH** variable but first substituting **manual** for **tcl**.

help-on-widget will look for a file of the form **$taskname.$widget.fmt** or **$taskname/$widget.fmt** and **help-on-task** will look for **$taskname.fmt** or **$taskname/$taskname.fmt** or **$taskname/fmt**

where **fmt** can be one of **html dvi ps prn txt**

The latter examples allow all help for a task to be within a directory having the name of the task (or the specified alternative).

example - for the task **ap** (apparatus access) manual information on the **Claim** command can be found in the file **/MIDAS/manual/ap.CLAIM.txt** and the **Help** button shows information in the file **/MIDAS/manual/ap.txt**

Note - since the task **help** information probably contains information from the individual widget files records of the form **+++widget.txt** are permitted in the **task.txt** file and cause the contents of the file **task.widget.txt** to be substituted.

NOTES

If an action calls the **help** task which then cannot find any information the task level actions are then tried as a secondary source.

Also if the task level action for a **widget** is invoked but no help is available then the **help** task is tried as a secondary source.

All procedures used at task level to implement the **Help System** are in the **common.tk.tcl** file. The necessary bindings are included within the **:frame** and individual **widget** procedures.

The **Alt+F1** binding is available to aid implementors check new and updated level 1 help. The binding calls the **:load-help** procedure (which is also called when the task loads the **common.tk.tcl** file) to update the local help information held within the task.

Useful Procedures

procedure self

The name of the task (which should also be the name of the parent frame **widget**) should be set in the global variable **framename** or **frame**. By default the variable **frame** is set to the task name.
The frame title (which will be written by the window manager in the window title bar) should be set in the global variable **frametitle** or **title**.
The icon title (which will be used to label the icon when the window is closed) should be set in the global window **icontitle** or **iconname**.
The procedure **self** may have the argument **frame-name** which returns the frame name; the argument **frame-title** which returns the frame title or the argument **icon-title** which returns the icon title.

procedure click

The call-back mechanism from the widgets requires that a click procedure exist. The library contains a default click procedure which may be used if the rules enforced by it are acceptable.
The frame name containing the widget is stored in the TCL global variable **FRAME**; the command returned is stored in the TCL global variable **COMMAND** and the widget returned is stored in the TCL global variable **WIDGET.**
If the **COMMAND** is **:quit** then the procedure **click-quit** is called with **FRAME** as an argument.
If the call-back is from **WIDGET** then an attempt is made to call the procedure **click-widget** if it exists. The number of argument in the supplied procedure is checked. Zero or one argument (to hold returned value) are acceptable. If no procedure with the name **click-widget** is found then an attempt is made to invoke the procedure **click-default**. Again zero or one arguments are acceptable. If no procedure with the name **click-default** is found then the call-back is logged to the diagnostic stream and ignored.

procedure click-quit

Accepts zero or one argument (a frame widget name). The frame is iconized. The default frame is the top level frame from the application.

procedure start

If the application is new (determined by the value of the TCL global variable **started**) the procedure first calls the procedure **make-frame** otherwise it deiconizes the frame and brings it to the front on the screen. The procedure **click-redisplay** is called if it exists.

procedure finish

Destroys the frame and sets the TCL global variable **started** to its initial value.

A number of procedures are available which can be used to control frame widgets. By default if a frame widget name is not supplied then they act on the parent frame for the task.

procedure open-frame (also **bring-to-front** which is an alias)

Accepts an optional argument which is a frame widget name. Causes the frame to be deiconified if it is closed and the footer width to be configured using the width of the displayed frame.

procedure show-frame

As open-frame except that the footer width is not configured. Should be used if the width of the frame may be unreliable - ie immediately after creation.

procedure close-frame

Accepts an optional argument which is a frame widget name.

procedure destroy-frame

Accepts an optional argument which is a frame widget name.

procedure centre-frame

Positions the frame in the center of the screen. Accepts an optional argument which is a frame widget name.

procedure position-frame x y [frame]

Positions the frame at the screen screen co-ordinates **x y**. Accepts an optional argument which is a frame widget name.

Procedures which generate general purpose dialog or transient pop-up windows.

procedure midas_dialog

This procedure accepts the same arguments as **tk_dialog** which it replaces. It contains a number of implementation changes to circumvent problems in **tk_dialog**. The global variable **dialog_wraplength** (which has the default value 6I) controls the width of the dialog window.

procedure midas-warning *msg*

Creates a warning dialog pop-up window containing the text *msg* and a button labelled **continue**. The procedure returns only after a mouse click on the **continue** button.

procedure midas-report *msg*

Creates a report dialog pop-up window containing the text *msg* and a button labelled **Ok**. The procedure returns only after a mouse click on the **Ok** button.

procedure midas-confirmation *msg*

Creates a confirmation dialog pop-up window containing the text *msg* and buttons labelled **cancel** and **continue**. The procedure returns only after a mouse click on one of the buttons. The procedure returns the value **0** if the **cance**l button is selected and the value **1** if the **continue** button is selected.

procedure midas-error *error msg*

If *error* is not a null string then it is an error code returned by a **Midas** library routine. An error report is constructed using *error,* any text associated with *error* and *msg*. An error report dialog pop-up window containing the text generated and buttons labelled **continue** and **abort** is created. The procedure returns only after a mouse click on one of the buttons. The procedure returns the value **0** if the **continue** button is selected and the value **1** if the **abort** button is selected.

procedure display-midas-error-report ***p*** *error msg*

If *error* is not a null string then it is an error code returned by a **Midas** library routine. An error report is constructed using *error,* any text associated with *error* and *msg*. An error report dialog pop-up window containing the text generated and buttons labelled **continue** and **Stack Trace** is created. The procedure returns only after a mouse click on one of the buttons. The procedure returns the error report text. Note that the first argument (p) is no longer used.

procedure remove-midas-error-report ***args***

Retained for compatibility with previous versions. Has no current action.

procedure midas-show-file ***file***

Displays the contents of the file having path *file* in an information frame containing a text scroll window.

procedure midas-information ***text***

Displays the contents of the argument *text* in an information frame containing a text scroll window. This procedure should be used in preference to **midas-report** and **midas-warning** when the amount of text may be excessive for insertion in a dialog pop-up window which cannot be scrolled.

procedure midas-save-window ***name***

Normally called by the default binding to the **Control+S** keyboard sequence. Permits the contents of a **text** or **listbox** window to be saved to a disc file. A browser window is started to obtain the filename to be used.

procedure midas-print-window ***name***

Normally called by the default binding to the **Control+P** keyboard sequence. Permits the contents of a **text** or **listbox** window to be printed.