Calendar

The calendar metawidget provides a calendar that allows for selection of a date by using the mouse or the keyboard. The user can scroll through the days and weeks by using the cursor keys, and through the month and years with two arrow buttons and the control key. A double-click on the date display field sets the current day.

Options

-**controls top**|**bottom**
-**font** font-spec

Commands

set ?date-string?
get ?format-string?
scroll integer ?**days**|**months**|**years**?
bind bind-arguments
headerconfig option-value pairs
Applies all option-value pairs to the calendar's headline, showing the weekdays. Accepts all options that can be applied to a text widget tag.
headercget option

Combobox

The combobox metawidget is a combination of an entry widget and a listbox.

Options

-**lines** integer
-**entries** arg arg ...
-**state normal**|**disabled**|**restricted**
-**command** command-string
Executes the specified command-string whenever a new value is picked from the list.

Commands

see ?index?

Document

One windows in MDI application window.

Options

-**title** string
-**image** image-name
-**icontext** string
-**x**|-**y**|-**width**|-**height** pixels
-**font** font-spec
-**foreground**|-**background** color
-**state normal**|**minimized**|**maximized**|**withdrawn**
-**minsize** {width height}

Commands

menu options
Modifies the pull-down menu that pops up below the document window's icon, when clicked on the icon. The options can be all option-value pairs that the standard menu widget accepts. This command allows to add or disable entries to the menu.
lower|**raise**
pack|**grid**|**place** widget args
Arrange horizontally|**vertically**|**cascade**|**tile**|**icons**|**maximize**|**minimize**

Gridcontrol

The gridcontrol metawidget is a rather complex widget and can be used to display data in rows and columns, hence in table format. Columns are addressed by name, rows by number. Selection is supported in the regular way. Cells and cell ranges can be tagged similar to text in a text widget. By configuring the tags, cells can change their color, font etc. Columns can be resized with proper text clipping.
Some commands require to specify a cell or a cell range. This is done in the format column-name.row-number. E.g. salary.4 specifies the cell in the fourth row of the salary column. Wildcard are also allowed: lastname.* specifies all cells of the column lastname, *.12 all cells of row 12. Consequently, *.* identifies the entire table.

Options

-**font** font-spec
-**update full**|**partial**|**none** Controls how a column is updated during a resize operation.
-**selectmode single**|**multiple**
-**onselect** command-string
If command-string is not empty, then it is evaluated each time the selection changes.

Commands

column insert|**delete**|**configure**|**cget**|**fit**|**bind**|**names**|**exists** ?args?
column insert column-name position ?options?
column delete column-name
column configure column-name ?options?
There are three special options:
-**width** pixels
-**align left**|**right**|**center**|**numeric**
-**text** string
Otherwise, all options are accepted that a text widget tag would accept
column cget column-name option
column fit column-name
column bind column-name ?bind-args?
column names
column exists column-name
insert row-number ?value-list ...?
Inserts one or more rows, starting at the specified row number, and fill them up with values.
delete from-row ?to-row?
set cell-spec ?value-list ...?
get ?cell-spec1? ?cell-spec2?
tag add|**delete**|**remove**|**cget**|**configure**|**lower**|**raise**|**ranges**|**names**|**bind** ?args?
This command allows to apply and control tags for cells and cell ranges.
tag add tag-name cell-spec1 ?cell-spec2?
tag remove tag-name cell-spec1 ?cell-spec2?
tag names cell-spec
tag ranges tag-name
returns all cells that have the specified tag currently attached.
All other tag commands work exactly like the tag commands for the text widget. In particular, the selection can be set by using the special tag "**sel**".
bind bind-arguments
see cell-spec
rows
Returns the total number of rows.

Ibutton

The ibutton metawidget is just like a regular button, except it can have an image inside it.

Options

-**image** image-name
Specifies the image to be displayed.

Iconbox

The iconbox metawidget acts as a container for icons. An icon consists of an image and a label and usually represent some object (e.g. a file). Hence, attributes can also be displayed for each icon.

Options

-**font** font-spec Sets the font that is used for the label of the icons.
-**update full**|**partial**|**none** during a resize operation
-**selectmode single**|**multiple**
-**onselect** command-string
-**view large**|**small**|**list**
-**columns** column-spec-list
{ Caption ?alignment ?width ?minimal_width??? }
alignment: **l** (left), **r** (right), **c** (center) or **n** (numeric)
width and minimal is be specified in pixels.

Commands

insert position icon-name ?options?
position: integer, or "end"
delete ?icon-name ...?
If no icon names are given, then all icons are deleted.
iconconfigure icon-name option value ?option value ...?
-**text** string: The label to be used for the icon
-**image** image-name: The image to be used when in "large icons" view
-**smallimage** image-name: The image to be used when in "small icons" and list view
-**values** list: The attributes to be displayed when in list view
-**user** user-data: Arbitrary user data. Just stored with the icon but never used
All other options are applied to the icon's label (-font, -bg etc.).
iconcget icon-name option
iconbind icon-name bind-arguments
selection clear|**get**|**set** ?icon-names-list?
sort ?-**column** column-caption? ?lsort-options?
see icon-name
bind bind-arguments
size Returns the number of icons in the iconbox.
names Returns a list with the names of all icons in the iconbox.

Listcontrol

Multicolumn listbox. Columns can be resized by dragging them. An icon can be placed into each column for each row individually. Cells or cell ranges can be tagged similar like with the gridcontrol widget.

Options

-**font** font-spec individual rows can be controlled by tag
-**update full**|**partial**|**none** how a column is updated during a resize operation
-**selectmode single**|**multiple**
-**onselect** command-string

Commands

column insert|**delete**|**configure**|**cget**|**fit**|**bind**|**names**|**exists** ?args?
column insert column-name position ?options?
column configure column-name ?options?
-**width** pixels specifies the column's width
-**align left**|**right**|**center**|**numeric** controls the alignment of text inside the column
-**text** string specifies the column heading
-**minsize** pixels specifies the column's minimal width
Otherwise, all options are accepted that a text widget tag would accept.
column cget column-name
column fit column-name resizes a column to best width
column bind column-name ?bind-args? accepts arguments like the bind command
column names returns a list of all currently defined columns
column exists column-name returns 1 if a column exists, 0 otherwise.
insert row-number ?value-list ...?
delete from-row ?to-row?
set ?-**columns** column-list? start-row ?value-list ...?
get ?-**columns** column-list? first-row ?last-row?
tag add|**delete**|**remove**|**cget**|**configure**|**lower**|**raise**|**ranges**|**names**|**bind** ?args?
tag add tag-name cell-spec1 ?cell-spec2? **tag names** cell-spec
tag remove tag-name cell-spec1 ?cell-spec2? **tag ranges** tag-name
selection clear|**get**|**set** ?row-range ...?
image create|**delete**|**configure**|**cget**|**names** ?cell-spec? ?image-options?
Allows to attach an image to a particular cell or cell range. The image options must be those as accepted by the image command of the text widget.
bind bind-arguments
see row
size
Returns the total number of rows.

Pane

Container for two user-defined widgets.

Options

-**orientation vertical**|**horizontal**
-**width** pixels Sets the width of the draggable frame.
-**resize both**|**first**|**second**
-**update** boolean if the two user widgets shall be redrawn during dragging

Commands

set position ?**both**|**first**|**second**?
Sets the pane to a new position. **If both** (the default) is specified, position is taken as a **percentage** value (e.g. set 50 would center the pane). **Otherwise as a pixel** value.
get ?**both**|**first**|**second**?
place|**pack**|**grid first**|**second** window-name ?args?
Places, packgs or grids a window either into the first (left, top) or second (right, bottom) part of the container widget.

Progressbar

The progressbar metawidget allows to visualize the progress of some process, e.g. the loading of a file or processing of data. The default widget is the area behind the progress bar and is of type frame.

Options

-**background** color
-**foreground** color
-**font** font-spec
-**format** format-string
Defines the format to be used for the text display.

Commands

set value ?base-value?

Scrollbox

The scrollbox metawidget is identical to a regular listbox, except it has a vertical and a horizonal scrollbar attached that pop up only when they are needed. They allow to address entries in the listbox by their name (i.e. the text that is displayed) rather than their index.

Options

-**scrollbar auto**|**on**|**off**

Commands

find entry ?entry ...?
selection setbyval|**get**|**standard**-options ?args?
Extents the standard listbox selection command by two functions:
set entry ?entry ...?
Deletes all existing entries and inserts new ones.
remove entry ?entry ...?
Removes particular entries from the list.
sort standard-lsort-args

Spinentry

Entry widget with two arrow buttons on the right side

Options

-**step** value
-**speed** milliseconds
-**minimum**|-**maximum** value
-**onerror** cmd
Defines a script to be called when the value in the entry widget cannot be incremented respectively decremented. If cmd is an empty string, then errors will not be caught.
-**value** value
Sets the value inside the entry widget. Also checks if the value is of numeric type. If not, an error is returned.

Statusbar

The statusbar metawidget sticks to the bottom of a window and displays some text.

Options

-**ticks** number — Number of ticks on progress bar
-**progress** value — The value must be between 0 and 1.
-**state normal**|**withdrawn** — Shows or hides the status bar.
-**text** string — Sets a text in the status bar.

Commands

push string — set new bar text and save previous
pop — return previous bar text
add field-name ?args?
delete field-name ?field-name ...?
itemconf field-name option value ?option value ...?
itemcget field-name option

Configures a field. Each field consists of a frame, an image and a text label. Consequently, all options are accepted that these three widget types would accept.

Tabcontrol

The tabcontrol metawidget is a container widget that contains a row of tabs at the top.

Options

-**width auto**|pixels|percent%

Commands

insert tab-name row ?args?
delete tab-name
tabconfigure tab-name option value ?option value ...?

Configures a tab. Each tab is actually a label, therefore all label options are accepted.
In addition there are two special options:
-**window** specifies the user widget to be displayed when the tab is active
-**command** specifies a script to be evaluated each time the tab is activated

tabcget tab-name option
invoke — Activates a tab and shows its associated user widget.
get ?**active**|**row**?

If no arguments is specified, returns all tab names. If active is specified, returns the active tab. If a valid row number is specified, returns the tabs in this row.

bind tab-name bind-arguments

Textframe

The textframe metawidget is a regular frame widget with a title text

Options

-**anchor n**|**s**|**w**|**e**|**c**
-**offset** value

Sets an offset value for the title text. There are three different formats: E.g. 20% sets the label 20% of the frame's width off from the left. 20 sets the label 20 pixel from the left border (good with -anchor w). end-20 sets it from the frame's right border (good with -anchor e).

-**font** font-spec
-**text** string

Commands

pack|**place**|**grid** window ?args?

Toolbar

Toolbars contain a number of icons that represent entries from the application's menu.

Options

-**side top**|**bottom**|**left**|**right**
-**state normal**|**fixed**|**withdrawn**

Commands

add button|**checkbutton**|**radiobutton**|**separator** name ?args?

For the radiobutton type, there is one special option -**group** group-name

delete name ?name ...?
itemconf name option value ?option value ...?

Configures an item. Accepts all options that a button widget would accept. In particular, -**image** to set an icon and -**command** to define an action are important here.

itemcget name option
invoke name ?new-state-boolean?
set name ?new-state?

Same as invoke except that the item's associated command is not called.

get name

Returns the state of a tool button as boolean. For regular buttons, always returns 0.

names ?pattern?

Tooltip

Options

-**delay** milliseconds

Commands

add widget-name string

Activates the tooltip feature for a particular user-defined widget and defines the text to be displayed. The text can be changed by calling this command again without having to call forget before.

forget widget-name

Removes the tooltip feature for a particular widget.

Treecontrol

The treecontrol metawidget allows to built an Explorer-style tree with an arbitrary number of nodes.

Options

-**text** string

label of the root node.

-**image** image-name

Sets the image to be used for the root node.

-**fullexpand** boolean

Controls whether automatic expanding of all subnodes of a node shall be allowed, when the Shift key was pressed by the user.

-**selectmode single**|**multiple**
-**onexpand** command

The command specified herein is evaluated whenever a node is being expanded and the children of that node are not known yet, or the node is not defined as "final" (see nodeconfigure below).

-**onselect** command

Commands

insert node ?args?
delete ?node?
nodeconfigure node option value ?option value ...?

-**parent** node: Sets a new parent node. The node will be drawn as a child of that parent node. The tree's root node is represented by an empty string.
-**text** string: Sets the nodes label.
-**image** image-name: Sets an icon for the node. Empty string, to removes the icon.
-**final** boolean: Tells the tree wigdet that the given node is complete.
-**user** string: Arbitrary user data. Just stored with the node but never used.

nodecget node option

-**children** returns a list of all child nodes.
-**indention** returns the current indention level of the node.
-**expanded** returns a boolean indicating if the node is currently expanded.
-**tags** returns a list of all tags currently associated with the node
-**visible** returns a boolean indicating if the node is visible, i.e. all of its parents are expanded. (The node can still be not on the screen because the tree's viewport is somewhere else.)

nodebind node ?bind-arguments?
expand node ?recursive-flag?
collapse node ?recursive-flag?
move node ?-**byname**? ?-**after**? position
tag add|**delete**|**remove**|**cget**|**configure**|**lower**|**raise**|**names**|**nodes**|**bind** ?args?
selection clear|**get**|**set** ?node-names-list?
get from-row ?to-row? — Returns a list of nodes that are between the two specified rows.
see node
bind bind-arguments
nodes — Returns an unordered list of all nodes, whether visible or not.
tree node — Returns the structure of the subtree as lis.

Window

The window metawidget's purpose is to unify the window widget and the wm command. It is in so far not a true metawidget, but it makes the handling of toplevel widgets easier.

Options

-**geometry** geometry-string
-**minsize**|-**maxsize** list
-**resizable** { allow-x allow-y }
-**override** boolean — Controls if the window is ignored by the window manager.
-**ondelete** command
-**title** string
-**state normal**|**iconic**|**withdrawn**

Commands

resize width height
move x y
protocol wm-protocol-args

mkWidgets 1.3

http://mkextensions.sourceforge.net/
quickcard ver. 0.1 by michael.heca@email.cz

THE METAWIDGET COMMAND

metawidget create ClassName InitProc ExitProc ?-**type** WidgetType? ?-**default** WidgetName? ?-**command** CommandName? ?args?

This command creates a new metawidget class with the name ClassName.

metawidget proc ClassName ProcName Args Body

This command defines a procedure that is associated with a metawidget. The automatic variable $this represents the current metawidget.

metawidget command ClassName CommandName ProcName

This defines a metawidget command and associates it with a metawidget procedure.

metawidget option ClassName OptionName ?SetProc? ?GetProc?

This defines a metawidget option.

metawidget delete ClassName

The delete subcommand deletes a metawidget class.

metawidget names

This simply returns all currently defined metawidgets.

metawidget info ClassName **procs** | **commands** | **options**

This returns the procedures, commands and options defined for the specified class.

metawidget export ClassName

The code as generated by the Metawidget package is returned.

SPECIAL COMMANDS

my WidgetVar ?NewValue?
Sets or retrieves a widget-variable.

unmy WidgetVar ?WidgetVar ...?
Unsets widget-variables.

myarray option WidgetArray ?args?
This command is just like the standard "array" command, except that it applies to widget-arrays.

myinfo class
myinfo vars
myinfo exists Widget|Var
Either returns the metawidget's class name, or all currently defined widget-variables, or a flag if the specified WidgetVar exists.

our ClassVar ?NewValue?
Sets or retrieves a class-variable.

unour ClassVar ?ClassVar ...?
Unsets class-variables.

ourarray option ClassArray ?args?
This command is just like the standard "array" command, except that it applies to class-arrays.

ourinfo vars
ourinfo exists ClassVar

TOOL COMMANDS

mkw.lassign List ?VarName ...?
Assigns all elements in List to the given VarNames from left to right.

mkw.lextend List ?Value ...?
If Value is not yet an element of List, Value is appended to the list and the extended list is returned. Otherwise, the command simply returns List.

mkw.lshrink List ?Value ...?
If Value is an element of List, Value is deleted from the list and the reduced list is returned. Otherwise, the command simply returns List.

mkw.lchange List OldValue NewValue
If OldValue is an element of List, OldValue is replaced by NewValue and the modified list is returned. Otherwise, the command simply returns List.

mkw.decode Expr List ?DefaultValue?
This is similar to "string map", which is not available in Tcl 8.0.

mkw.complete Expr List
Expr is compared against each of the elements in List to find a match.

mkw.options List ?OptionSpec ...?
This is a simple option processing routine. options can have no arguments. OptionSpec is simply the name of the option, e.g. -dictionary. For options with a value, an OptionSpec consists of two or three elements: The first is the option name, the second either a list of allowed values or an asterisk * for any value. The third element is optional and specifies a default value if the option is not found in List.
For each option found, a variable with the same name as the option is set in the calling procedure's context.

Aclock

The aclock metawidget resembles an analog clock, with an hour and a minute pointer. The pointers can be dragged with the mouse. The clock can also be set with two arrow buttons and a time entry field. A double-click on the gray area between the buttons and the time entry field sets the current time.

Options

-**controls top**|**bottom**
-**format** format-string

Accepts a format string like for the standard clock format command.

-**font** font-spec

Commands

set ?time-string?
get
scroll minutes

Sets the clock's time forward or backward according to the specified minutes.

bind bind-arguments