Gadget

COLLABORATORS			
	TITLE : Gadget		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		July 31, 2024	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

Gadget

1.1 Gadget

```
PureBasic Gadget library

  Gadgets in PureBasic are based on the AmigaOS 'GadTools' library
  and provide a very easy way to setup the layout. All the gadgets
  types are supported.

Commands summary:

   ActivateGadget
   ButtonGadget
   CheckBoxGadget
   CreateGadgetList
   CycleGadget
   DisableGadget
   FreeGadget
   GadgetBevelBox
   GetGadgetState
   GetGadgetText
   InitGadget
   IntegerGadget
   ListViewGadget
   NoGadgetBorder
   NumberGadget
   OptionGadget
   PaletteGadget
   RefreshGadget
   ScrollerGadget
   SetGadgetAttribute
   SetGadgetFont
   SetGadgetState
   SetGadgetTagList
   SetGadgetText
   SliderGadget
   StringGadget
   TextGadget


Examples:
```

```
    Gadget Demo
```

1.2 activategadget

```
  SYNTAX
Result = ActivateGadget(#Gadget)

  FUNCTION
Cause the specified #Gadget to be activated, but only if
its a string or integer gadget.  Result will show if the
activation have taken place, TRUE if it have and FALSE if
its not. And the gadget can't be disabled.

Note:
====
The #Gadget will only be activated if these set of
conditions are met, only to ensure that user won't
be interrupted.

* The window where the gadget is must be active.
* No other gadgets may be in use, including
  system gadgets.
* The right mouse button (ie. menus) cant be
  held down.
```

1.3 buttongadget

```
  SYNTAX
ButtonGadget(#Gadget,x,y,Width,Heigth,Text$)

  COMMAND
Create a button gadget on the current window. #Gadget will be
the number returned by EventGadgetID(). The x and y will set
the position, x=0 and y=0 will put the gadget at topleft in
client area. Width and Heigth set the dimension of the gadget
and Text$ supply the text that will be put into the gadget,
set it to zero if no text is wanted.

Available tags that could be passed with SetGadgetTagList()
to this command, if the default settings need to be changed.

 #GA_Immediate (BOOL)
 --------------------
 Hear #IDCMP_GADGETDOWN events from button gadget (defaults is
 FALSE). (V39)
```

1.4 checkboxgadget

```
  SYNTAX
CheckBoxGadget(#Gadget,x,y,Width,Heigth,Text$)

  COMMAND
Create a checkbox gadget in the current window. #Gadget will
be the number returned by EventGadgetID().  The x and y will
set the position, x=0 and y=0 will put the gadget at topleft
in client area.  Width and Heigth set the dimension of the
gadget and Text$ is an optional label that will be put at
right side of the checkbox, but if its not wanted just set
it to zero.
  _
 |_| Enable automatic saving.


Available tags that could be passed with SetGadgetTagList()
to this command, if the default settings need to be changed.

  #GTCB_Scaled (BOOL)
  ------------------
  If true, then checkbox imagery will be scaled to fit the gadget's
  width & height.  Otherwise, a fixed size of CHECKBOXWIDTH by
  CHECKBOXHEIGHT will be used. (defaults is FALSE)  (V39)
```

1.5 creategadgetlist

```
  SYNTAX
Result = CreateGadgetList()

  FUNCTION
Use this function to make it possible to attache
gadgets to the current window.
```

1.6 cyclegadget

```
  SYNTAX
CycleGadget(#Gadget,x,y,Width,Heigth,Text$,Labels())

  COMMAND
Create a cycle gadget on the current window. #Gadget will be
the number returned by EventGadgetID(). The x and y will set
the position, x=0 and y=0 will put the gadget at topleft in
client area. Width and Heigth set the dimension of the gadget
and Text$ supply an optional label to be put at the left, set
it to zero if no label is wanted.  The Labels() parameter that
will be the content of the cycle gadget is an array of strings
and the last one  must  not contain any characters, as this one
define that no more labels is to be expected.

Dim CycleGadgetItems.s(3)
  CycleGadgetItems(0) = "Item1"
```

```
  CycleGadgetItems(1) = "Item2"
  CycleGadgetItems(2) = "Item3"

                      ----------
  Choose your Item: | | Item1  |
                      ----------
```

There is no available tags for this command as all possible settings is supported within this or other functions.

1.7 disablegadget

```
  SYNTAX
DisableGadget(#Gadget,State)

  STATEMENT
```

Disable or enable a gadget. If State = 1, the gadget will be disable, if State = 0 then it will be enabled.

1.8 freegadget

```
  SYNTAX
FreeGadget(#Gadget)

  STATEMENT
```

Cause the specified #Gadget to be freed, but if -1 is used then its all the gadgets on the current window that will be freed.

1.9 gadgetbevelbox

```
  SYNTAX
GadgetBevelBox(x,y,Width,Height,Style)

  STATEMENT
```

Draw a bevelbox in the current window at the specified position and with specified dimension. Style could be either raised 0 or recessed 1.

1.10 getgadgetstate

```
  SYNTAX
Result = GetGadgetState(#Gadget)

  FUNCTION
```

Get the state of any gadget at any time, button, string and text gadgets will always return zero.

1.11 getgadgettext

```
  SYNTAX
Text$ = GetGadgetText(#Gadget)

  FUNCTION
Return the text content of a string or text gadget,
other gadget types will always return "".
```

1.12 initgadget

```
  SYNTAX
Result = InitGadget(#MaxGadgets)

  COMMAND
Initialize the gadget environment. You must put this command
before using any of the other gadget function and also check
Result if you want to be sure its alright.
```

1.13 integergadget

```
  SYNTAX
IntegerGadget(#Gadget,x,y,Width,Heigth,Text$,Content)

  COMMAND
Create a integer gadget in the current window. #Gadget will
be the number returned by EventGadgetID(). The x and y will
set the position, x=0 and y=0 will put the gadget at topleft
in client area.   Width and Heigth set the dimension of the
gadget and Text$ is an optional label that will be put at
the left, but if its not wanted just set it to zero. Content
is the number that will be put in the gadget.


Available tags that could be passed with SetGadgetTagList()
to this command, if the default settings need to be changed.

 #GA_Immediate (BOOL)
 --------------------
 Hear #IDCMP_GADGETDOWN events from integer gadget (defaults
 to FALSE). (V39)

 #GA_TabCycle (BOOL)
 -------------------
 Set to TRUE so that pressing <TAB> or <Shift-TAB> will activate
 the next or previous such gadget. (defaults to TRUE,  unlike
  regularIntuition string gadgets which default to FALSE).  (V37)

 #GTIN_MaxChars (UWORD)
 ----------------------
 The maximum number of digits that the integer gadget is to hold
 (defaults to 10). (V36)
```

```
#GTIN_EditHook (struct Hook *)
------------------------------
Hook to use as a custom integer gadget edit hook (StringExtend->
EditHook) for this gadget. GadTools will allocate the
StringExtend->WorkBuffer for you.
(defaults to NULL). (V37)

#STRINGA_ExitHelp (BOOL)
------------------------
Set to TRUE to have the help-key cause an exit from the integer
gadget.  You will then receive an #IDCMP_GADGETUP event with
Code = 0x5F (rawkeyfor help). (defaults to FALSE) (V37)

#STRINGA_Justification
----------------------
Controls the justification of the contents of an integer gadget.
Choose one of STRINGLEFT, STRINGRIGHT, or STRINGCENTER  (defaults
to STRINGLEFT). (V37)

#STRINGA_ReplaceMode (BOOL)
---------------------------
If TRUE, this integer gadget is in replace-mode (defaults to FALSE
(insert-mode)). (V37)
```

1.14 listviewgadget

```
  SYNTAX
ListViewGadget(#Gadget,x,y,Width,Heigth,Text$,ListBase)

  COMMAND
Create a listview gadget in the current window. #Gadget will
be the number returned by EventGadgetID().  The x and y will
set the position, x=0 and y=0 will put the gadget at topleft
in client area.   Width and Heigth set the dimension of the
gadget and Text$ is an optional label that will be put above
it, but if its not wanted just set it to zero. ListBase is the
pointer given by the ListBase() function and its a linked list,
created with NewList, with a structure type like this one:

  Structure ListView
    Pad.w     ; Skip these two first bytes.
    String.s  ; The string to display in the listview.
  EndStructure


    List Content:
 ____________________ _
|Item 1              | |
|Item 2              | |
|Item 3              |_|
|                    |_|
|____________________|_|
```

```
Available tags that could be passed with SetGadgetTagList()
to this command, if the default settings need to be changed.

 #GTLV_Top (WORD)
 ---------------
 Top item visible in the listview.  This value will be made
 reasonable if out-of-range (defaults to 0). (V36)

 #GTLV_MakeVisible (WORD)
 -----------------------
 Number of an item that should be forced within the visible
 area of the listview by doing minimal scrolling.  This tag
 overrides #GTLV_Top. (V39)

 #GTLV_ReadOnly (BOOL)
 --------------------
 If TRUE, then listview is read-only (defaults to FALSE). (V36)

 #GTLV_ScrollWidth (UWORD)
 ------------------------
 Width of scroll bar for listview.  Must be greater than zero
 (defaults to 16). (V36)

 #LAYOUTA_Spacing (UWORD)
 -----------------------
 Extra space to place between lines of listview (defaults to 0).
 (V36)

 #GTLV_ItemHeight (UWORD)
 -----------------------
 The exact height of an item. This is normally useful for listviews
 that use the #GTLV_CallBack rendering hook (defaults to ng->ng_TextAttr->
 ta_YSize). (V39)

 #GTLV_CallBack (struct Hook *)
 -----------------------------
 Callback hook for various listview operations. As of V39, the
 only callback supported is for custom rendering of individual
 items in the listview. The call back hook is called with:

  A0 - struct Hook *
  A1 - struct LVDrawMsg *
  A2 - struct Node *
  The callback hook *must* check the lvdm_MethodID field of the
  message and only do processing if it equals LV_DRAW. If any
  other value is passed, the callback hook must return LVCB_UNKNOWN

 #GTLV_MaxPen (UWORD)
 -------------------
 The maximum pen number used by rendering in a custom rendering
 callback hook. This is used to optimize the rendering and scrolling
 of the listview display (default is the maximum pen number used by
 all of TEXTPEN, BACKGROUNDPEN, FILLPEN, TEXTFILLPEN, and BLOCKPEN.
 (V39)
```

1.15 nogadgetborder

```
  SYNTAX
NoGadgetBorder(#Gadget)

  STATEMENT
Must be put after a gadget declaration and will remove the
border around the specified gadget ie:

 ButtonGadget(1,10,10,100,100,"Hello",0)
 NoGadgetBorder(1)
```

1.16 numbergadget

```
  SYNTAX
NumberGadget(#Gadget,x,y,Width,Heigth,Text$,Content)

  COMMAND
Create a number gadget in the current window. The x and y
will set the position, x=0 and y=0 will put the gadget at
topleft in client area. Width and Heigth set the dimension
of the gadget and Text$ is an optional label that will be
put at the left, but if its not wanted just set it to zero.
Content is the number to put into the gadget.

This gadget is output only so user cant affect it so there
will be no return by GadgetEventID().

Available tags that could be passed with SetGadgetTagList()
to this command, if the default settings need to be changed.

 #GTNM_Border (BOOL)
 ------------------
 If TRUE, this flag asks for a recessed border to be placed
 around the gadget. (V36)

 #GTNM_FrontPen (UBYTE)
 ----------------------
 The pen to use when rendering the number (defaults to DrawInfo->
 dri_Pens[TEXTPEN]). (V39)

 #GTNM_BackPen (UBYTE)
 ---------------------
 The pen to use when rendering the background of the number
 (defaults to leaving the background untouched). (V39)

 #GTNM_Justification (UBYTE)
 ---------------------------
 Determines how the number is rendered within the gadget box. GTJ_LEFT
 will  make the rendering be flush with the left side of the gadget,
 GTJ_RIGHT will make it flush with the right side, and GTJ_CENTER will
 center the number within the gadget box. Under V39, using this tag
 also required using {#GTNM_Clipped, TRUE}, otherwise the text would
 not show up in the gadget. This has ben fixed in V40.
```

```
(defaults to GTJ_LEFT). (V39)

#GTNM_Format (STRPTR)
--------------------
C-Style formatting string to apply on the number before display.
Be sure to use the 'l' (long) modifier. This string is processed
using exec.library/RawDoFmt(), so refer to that function for
details. (defaults to "%ld") (V39)

#GTNM_MaxNumberLen (ULONG)
-------------------------
Maximum number of bytes that can be generated by applying the
#GTNM_Format formatting string to the number (excluding the NULL
terminator). (defaults to 10). (V39)

#GTNM_Clipped (BOOL)
-------------------
Determine whether text should be clipped to the gadget dimensions
(defaults to FALSE for gadgets without borders, TRUE for gadgets
with borders). (V39)
```

1.17 optiongadget

```
  SYNTAX
OptionGadget(#Gadget,x,y,Width,Heigth,Lables())

  COMMAND
Create a option gadget in the current window. #Gadget will
be the number returned by EventGadgetID(). The x and y will
set the position, x=0 and y=0 will put the gadget at topleft
in client area.   Width and Heigth set the dimension of the
gadget. Lables() parameter that represents the text that will
be put at right side of each knob is an array of strings and
the last one  must  not contain any characters, as this define
that no more lables is to be expected.

Dim OptionGadgets.s(3)
  OptionGadgets(0) = "January"
  OptionGadgets(1) = "February"
  OptionGadgets(2) = "March"
   _
  |_| January
   _
  |_| February
   _
  |_| March


Available tags that could be passed with SetGadgetTagList()
to this command, if the default settings need to be changed.

 #GTMX_Spacing (UWORD)
 --------------------
 The amount of space between each choice of a set of mutually
 exclusive gadgets.  This amount is added to the font height to
```

```
produce the vertical shift between choices (defaults to 1). (V36)

#GTMX_Scaled (BOOL)
------------------
If true, then mx gadget imagery will be scaled to fit the gadget's
width & height.  Otherwise, a fixed size of MXWIDTH by MXHEIGHT will
be used. When setting this tag to TRUE, you should typically set the
height of the gadget to be (ng.ng_TextAttr->ta_YSize + 1).
(defaults to FALSE.) (V39)

#GTMX_TitlePlace
----------------
One of PLACETEXT_LEFT, PLACETEXT_RIGHT, PLACETEXT_ABOVE, or PLACETEXT_BELOW,
indicating where the title of the gadget is to be displayed. Without this
tag, the  NewGadget.ng_GadgetText field is ignored for MX_KIND gadgets.
(V39)
```

1.18 palettegadget

```
  SYNTAX
PaletteGadget(#Gadget,x,y,Width,Heigth,Text$,Depth)

  COMMAND
Create a palette gadget on the current window. #Gadget will be
the number returned by EventGadgetID().  The x and y will set
the position, x=0 and y=0 will put the gadget at topleft in
client area. Width and Heigth set the dimension of the gadget
and Text$ supply an optional label to be put above the gadget,
set it to zero if no label is wanted. The Depth (ie: number of
colors) will be used to calculate how many color buttons the
palette gadget will contain.

   Choose your color:
 ____________________________
|__|__|__|__|__|__|__|__|
|__|__|__|__|__|__|__|__|


Available tags that could be passed with SetGadgetTagList()
to this command, if the default settings need to be changed.

#GTPA_ColorOffset (UBYTE)
-------------------------
First colour to use in palette (defaults to 0). (V36)

#GTPA_IndicatorWidth (UWORD)
----------------------------
The desired width of the current-colour indicator, if you want
one to the left of the palette. (V36)

#GTPA_IndicatorHeight (UWORD)
-----------------------------
The desired height of the current-color indicator, if you want
one one above the palette. (V36)
```

```
#GTPA_ColorTable (UBYTE *)
-------------------------
Pointer to a table of pen numbers indicating  which colors should
be used and edited by the palette gadget. This array must contain
as many entries as there are colors displayed in the palette gadget.
The array provided with this tag must remain valid for the life of
the gadget or until a new table is provided. (default is NULL, which
causes a 1-to-1 mapping of pen numbers). (V39)

#GTPA_NumColors (UWORD)
----------------------
Number of colors to display in the palette gadget. This override
#GTPA_Depth and allows numbers which aren't powers of 2. (defaults
to 2) (V39)
```

1.19 refreshgadget

```
  SYNTAX
RefreshGadget(#Gadget)

  STATEMENT
Cause the specified #Gadget to be refreshed, but if
-1 is used then its all the gadgets on the current
window that will be refreshed.
```

1.20 scrollergadget

```
  SYNTAX
ScrollerGadget(#Gadget,x,y,Width,Heigth,Text$,Total,Visible)

  COMMAND
Create a scroller gadget on the current window. #Gadget will be
the number returned by EventGadgetID(). The x and y will set the
position, x=0 and y=0 will put the gadget at topleft in client
area. Width and Heigth set the dimension of the gadget and Text$
supply an optional label to be put at left side of the gadget,
set it to zero if no label is wanted. Total represent the whole
amount of the scrollable area and Visible represent the amount
of the visible area.


Available tags that could be passed with SetGadgetTagList()
to this command, if the default settings need to be changed.

#GA_Immediate (BOOL)
-------------------
Hear every #IDCMP_GADGETDOWN event from scroller (defaults to
FALSE). (V36)

#GTSC_Top (WORD)
---------------
Top visible in area scroller represents (defaults to 0). (V36)
```

```
#GTSC_Arrows (UWORD)
--------------------
Asks for arrows to be attached to the scroller. The value supplied
will be taken as the width of each arrow button for a horizontal or
scroller, the height of each button for a vertical scroller (the
other dimension will match the whole scroller). (V36)

#PGA_Freedom
------------
Whether scroller is horizontal or vertical. Choose LORIENT_VERT or
LORIENT_HORIZ (defaults to LORIENT_HORIZ). (V36)
```

1.21 setgadgetattribute

```
  SYNTAX
SetGadgetAttribute(#Gadget,TagListID)

  STATEMENT
Change the attributes of the given #Gadget and that will
be done with the same tags that is specified for each
gadget type.
```

1.22 setgadgetfont

```
  SYNTAX
SetGadgetFont(FontID)

  STATEMENT
Sets the font which will be used by newly created gadgets.
```

1.23 setgadgetstate

```
  SYNTAX
Result = SetGadgetState(#Gadget,State)

  FUNCTION
Set a new state for the specified #Gadget, button, string
and text gadgets will not be affected but that could be
seen on Result as it is TRUE if the new state is set.
```

1.24 setgadgettaglist

```
  SYNTAX
SetGadgetTagList(TagListID)

  STATEMENT
```

```
Pass a taglist, to the next call, to any function
that create a gadget, like ButtonGadget() and more.
```

1.25 setgadgettext

```
  SYNTAX
SetGadgetText(#Gadget,Content$)

  STATEMENT
Change the text content of the specified #Gadget, that
is a string or text gadget, no other gadget types will
be affected but, that could be seen on Result when
its TRUE.
```

1.26 slidergadget

```
  SYNTAX
SliderGadget(#Gadget,x,y,Width,Heigth,Text$,Min,Max)

  COMMAND
Create a slider gadget on the current window. #Gadget will be
the number returned by EventGadgetID(). The x and y will set
the position, x=0 and y=0 will put the gadget at topleft in
client area. Width and Heigth set the dimension of the gadget
and Text$ supply an optional label to be at left of gadget,
set it to zero if no label is wanted. Min define the lowest
value and Max the highest value availeble.


Available tags that could be passed with SetGadgetTagList()
to this command, if the default settings need to be changed.

 #GA_Immediate (BOOL)
 --------------------
 If you want to hear each slider #IDCMP_GADGETDOWN event (defaults
 to FALSE). (V36)

 #GTSL_Level (WORD)
 ------------------
 Current level of slider (defaults to 0). (V36)

 #GTSL_MaxLevelLen (UWORD)
 -------------------------
 Maximum length in characters of level string  when rendered beside
 slider (defaults to 2). (V36)

 #GTSL_LevelFormat (STRPTR)
 --------------------------
 C-Style formatting string for slider level.  Be sure to use the 'l'
 (long) modifier.  This string is processed using exec.library/RawDoFmt(),
 so refer to that function for details. (defaults to "%ld"). (V36)
```

```
#GTSL_LevelPlace
----------------
One of PLACETEXT_LEFT, PLACETEXT_RIGHT, PLACETEXT_ABOVE, or PLACETEXT_BELOW,
indicating where the level indicator is to go relative to slider
(default to PLACETEXT_LEFT). (V36)

#GTSL_DispFunc ( LONG (*function)(struct Gadget *, WORD) )
----------------------------------------------------------
Function to calculate level to be displayed. A number-of-colors slider
might want to set the slider up to think depth, and have a (1 << n)
function here.  Defaults to none.  Your function must take a pointer
to gadget as the first parameter, the level (a WORD) as the second,
and return the result as a LONG. (V36)

#GTSL_MaxPixelLen (ULONG)
-------------------------
Indicates the maximum pixel size used up by the level display for any
value of the slider. This is mostly useful when dealing with proportional
fonts. (defaults to FontWidth*MaxLevelLen). (V39)

#GTSL_Justification (UBYTE)
---------------------------
Determines how the level display is to be justified within its alotted
space. Choose one of GTJ_LEFT, GTJ_RIGHT, or GTJ_CENTER (defaults to
GTJ_LEFT). (V39)

#PGA_Freedom
------------
Set to LORIENT_VERT or LORIENT_HORIZ to have a vertical or horizontal
slider (defaults to LORIENT_HORIZ). (V36)
```

1.27 stringgadget

```
  SYNTAX
StringGadget(#Gadget,x,y,Width,Heigth,Text$,Content$)

  COMMAND
Create a string gadget on the current window. #Gadget will be
the number returned by EventGadgetID(). The x and y will set
the position, x=0 and y=0 will put the gadget at topleft in
client area. Width and Heigth set the dimension of the gadget
and Text$ supply an optional label to be put on left side of
the gadget, set it to zero if no label is wanted. Content$ is
a string that will be put into the string gadget.

                  ____________________
 Enter your name: |____________________|


Available tags that could be passed with SetGadgetTagList()
to this command, if the default settings need to be changed.

 #GA_TabCycle (BOOL)
 -------------------
 Set to TRUE so that pressing <TAB> or <Shift-TAB> will activate the
```

```
next or previous such gadget. (defaults to TRUE, unlike regular Intuition
string gadgets which default to FALSE). (V37)

#GTST_EditHook (struct Hook *)
------------------------------
Hook to use as a custom string gadget edit hook (StringExtend->EditHook)
for this gadget. GadTools will allocate the StringExtend->WorkBuffer for
you. (defaults to NULL). (V37)

#STRINGA_ExitHelp (BOOL)
------------------------
Set to TRUE to have the help-key cause an exit from the string gadget.
You will then receive an #IDCMP_GADGETUP event with Code = 0x5F (rawkey
for help). (V37)

#STRINGA_Justification
----------------------
Controls the justification of the contents of a string gadget. Choose
one of #STRINGLEFT, #STRINGRIGHT, or #STRINGCENTER (defaults to #STRINGLEFT).
(V37)

#STRINGA_ReplaceMode (BOOL)
---------------------------
If TRUE, this string gadget is in replace-mode (defaults to FALSE
(insert-mode)). (V37)
```

1.28 textgadget

```
  SYNTAX
TextGadget(#Gadget,x,y,Width,Heigth,Text$,Content$)

  COMMAND
Create a text gadget in the current window.  The x and y
will set the position, x=0 and y=0 will put the gadget at
topleft in client area. Width and Heigth set the dimension
of the gadget and Text$ is an optional label that will be
put at the left, but if its not wanted just set it to zero.
Content$ is the text to be put into the gadget.

This gadget is output only so user cant affect it so there
will be no return by GadgetEventID().


Available tags that could be passed with SetGadgetTagList()
to this command, if the default settings need to be changed.

 #GTTX_CopyText (BOOL)
 --------------------
 This flag instructs the text-display gadget to copy the supplied text
 string, instead of using only pointer to the string.  This only works
 for the initial value of GTTX_Text set at CreateGadget() time. If you
 subsequently change GTTX_Text, the new text will be referenced by
 pointer, not copied.  Do not use this tag with a NULL GTTX_Text. (V37)

 #GTTX_Border (BOOL)
```

```
------------------
If TRUE, this flag asks for a recessed border to be placed around the
gadget. (V36)

#GTTX_FrontPen (UBYTE)
----------------------
The pen to use when rendering the text (defaults to DrawInfo->dri_Pens[TEXTPEN ↩
    ]). (V39)

#GTTX_BackPen (UBYTE)
---------------------
The pen to use when rendering the background of the text (defaults to
leaving the background untouched). (V39)

#GTTX_Justification (UBYTE)
---------------------------
Determines how the text is rendered  within the gadget box. GTJ_LEFT
will make the rendering be flush with the left side of the gadget,
GTJ_RIGHT will make it flush with the right side, and GTJ_CENTER will
center the text within the gadget box. Under V39, using this tag
also required using {GTNM_Clipped, TRUE}, otherwise the text would not
show up in the gadget. This has been fixed in V40. (defaults to GTJ_LEFT).
(V39)

#GTTX_Clipped (BOOL)
--------------------
Determine whether text should be clipped to the gadget dimensions
(defaults to FALSE for gadgets without borders, TRUE for gadgets
with borders). (V39)
```