Rexx/DW Reference

Version 2.1

Table of Contents

Table of Contents

Table of Contents

Table of Contents

1. RexxDW/Introduction [Modules]

[Top] [Modules]

DESCRIPTION

RexxDW is an external function package that allows a Rexx programmer to write cross platform GUI applications, using a lightweight portable GUI API called Dynamic Windows (dwindows).

USAGE

The fundamental building blocks of a RexxDW program consist of two major components; widgets and event handlers or callbacks.

Various widgets are packed into the required layout and events are connected to the widgets to carry out the required behaviour.

The structure of a RexxDW program consists of various initialisation; creation of the widgets, packing them, connecting callbacks, and then calling the event loop handler to dispatch the event to the appropriate callbacks. The callbacks are normal Rexx subroutines.

There are two event loop handlers depending on whether your Rexx interpreter supports the RexxCallback() API. DW_main() is used for interpreters that have RexxCallback(); DW_main_iteration() is used for those interpretes that don't.

DERIVED FROM

All of the functions in RexxDW are based on the same function in dwindows. Some function arguments will differ due to the nature of the Rexx API interface.

The format of the function names is dw_(target)_(action)[_(options)]. eg. for tree widgets, functions act on both the tree itself and the tree items within the tree widget, so DW_tree_new() executes the "new" action on the tree widget, DW_tree_item_change() executes the "change" action on the tree item, not the tree.

Some dwindows functions are not available in RexxDW, mainly because they are not able to be implemented: TBD, etc...

A number of functions have been added to RexxDW that are not part of dwindows: dw_and, dw_or, dw_radiobutton_set, dw_radiobutton_get,...

TODO

- document dw_exec()??
- add DW_container_get_column_type and DW_filesystem_get_column_type
- can dw_window_maximize() be implemented as dw_window_set_style() with MAXIMIZE attribute set and MINIMIZE attribute unset?
- no VALUE_CHANGE callback on spinbutton widget
- document that KEY_PRESS needs to be trapped in toplevel window but BUTTON_PRESS in low-level window (confirm on all platforms)

- DOCO for: dw_box_pack_at_index(box, item, index, width, height, hsize, vsize, pad) dw_box_unpack_at_index(box, index) dw_box_unpack(box)

BUGS

- bug in (Windows at least) item_select callback on container. Seems that the callback gets called twice irrespective of whether 0 or 1 is returned from the callback.
- bug (Windows) with drawing directly to a render window (ie not via a pixmap) (see rexxdw.rexx, Utility TAB)
- Under GTK+ 2.0 or greater, a call to dw_window_set_font() causes an expose event to fire. If dw_window_set_font() is called within the expose callback, you get an infinite loop. Bit of a bugger if your expose event sets different fonts for different lines!!!

PORTABILITY

RexxDW runs on Windows NT/2k/XP/Vista/W7, OS/2 3.0+, MacOS X, and on any Un*x platform that has GTK+ 1.x installed (but works better with GTK+ 2.x).

SEE ALSO

RexxDW lives at http://rexxdw.sf.net Dynamic Windows lives at http://dwindows.netlabs.org

2. RexxDW/Constants [Modules]

[Top] [Modules]

DESCRIPTION

The following "constants" are defined when RexxDW starts. By default, all constants are stored in an array with the stem preset to !REXXDW.! This can be changed by using the 'CONSTANTPREFIX' value of DW_variable(). If you use "Procedure" on your labels, you MUST "EXPOSE !REXXDW." or the stem you set with DW_variable() will not be visible. To reference the constants defined below, you must prefix them. So the "constant" DW_DESKTOP would be, by default, referenced in your code as !REXXDW.!DW_DESKTOP.

SEE ALSO

DW_variable

2.1. Constants/MiscellaneousFlags [Definitions]

[Top] [Constants] [Definitions]

NAME

MiscellaneousFlags

DESCRIPTION

The following is a list of miscellaneous flags.

ATTRIBUTES

- GTK_MAJOR_VERSION - 0 if NOT GTK+ platform, 1 for GTK+ 1.x, 2 for GTK+ 2.x
- HAVE_REXXCALLBACK - 1 if the interpreter supports RexxCallback() API
- DW_DESKTOP - the psuedo "window" which is the desktop
- DIRSEP - the Operating System directory separator
- PATHSEP - the Operating System path separator

2.2. Constants/FileDialogFlags [Definitions]

[Top] [Constants] [Definitions]

NAME

FileDialogFlags

DESCRIPTION

The following is a list of the pre-defined flags that are used in a call to DW_file_browse() to determine the dialog type displayed.

ATTRIBUTES

- DW_DIRECTORY_OPEN - a directory selector dialog
- DW_FILE_OPEN - a file open dialog
- DW_FILE_SAVE - a file save dialog

SEE ALSO

DW_file_browse

2.3. Constants/ContainerScrollFlags [Definitions]

[Top] [Constants] [Definitions]

NAME

ContainerScrollFlags

DESCRIPTION

The following is a list of the pre-defined flags that are used in a call to DW_container_scroll() to indicate the direction of scrolling.

ATTRIBUTES

- DW_SCROLL_UP - scroll up a number of lines
- DW_SCROLL_DOWN - scroll down a number of lines
- DW_SCROLL_TOP - scroll to the top of the container
- DW_SCROLL_BOTTOM - scroll to the bottom of the container

SEE ALSO

DW_container_scroll

2.4. Constants/WindowAttributeFlags [Definitions]

[Top] [Constants] [Definitions]

NAME

WindowAttributeFlags

DESCRIPTION

The following is a list of the pre-defined flags that are used in a call to DW_window_set_style() to change the behaviour of widgets as explained below.

ATTRIBUTES

The following work on all platforms:

```
- DW_DT_LEFT              - text in a widget is left-aligned
- DW_DT_RIGHT             - text in a widget is right-aligned
- DW_DT_CENTER            - text in a widget is centered
- DW_DT_CENTRE            - text in a widget is centred
- DW_DT_VCENTER           - text in a widget is centered vertically
- DW_DT_VCENTRE           - text in a widget is centred vertically
- DW_BS_NOBORDER          - remove the border from a BitmapButton
```

The following attributes only have an effect on OS/2:

```
- DW_DT_QUERYEXTENT
- DW_DT_UNDERSCORE
- DW_DT_STRIKEOUT
- DW_DT_TEXTATTRS
- DW_DT_EXTERNALLEADING
- DW_DT_TOP
- DW_DT_BOTTOM
- DW_DT_HALFTONE
- DW_DT_MNEMONIC
- DW_DT_WORDBREAK
- DW_DT_ERASERECT
```

SEE ALSO

DW_window_set_style

2.5. Constants/WindowStyleFlags [Definitions]

[Top] [Constants] [Definitions]

NAME

WindowStyleFlags

DESCRIPTION

The following is a list of the pre-defined flags that can be used when creating a new window. Any number of values from "Style Types" can be used plus optionally one value from "Initial State Types" are logically "or"ed together using DW_or().

ATTRIBUTES

Style Types:

```
- DW_FCF_TITLEBAR        - display titlebar
- DW_FCF_SYSMENU
- DW_FCF_MENU
- DW_FCF_SIZEBORDER
- DW_FCF_MINBUTTON       - display minimise button
- DW_FCF_MAXBUTTON       - display maximise button
- DW_FCF_MINMAX
- DW_FCF_DLGBORDER
- DW_FCF_BORDER
- DW_FCF_TASKLIST
- DW_FCF_NOMOVEWITHOWNER
- DW_FCF_SYSMODAL
- DW_FCF_HIDEBUTTON
- DW_FCF_HIDEMAX
- DW_FCF_AUTOICON
- DW_FCF_COMPOSITED
- DW_FCF_TEXTURED        - (only relevant to Mac) brushed metal background
```

Initial State Types:

```
- DW_FCF_MAXIMIZE        - start maximised
- DW_FCF_MINIMIZE        - start minimised
- DW_FCF_FULLSCREEN      - start full screen
```

SEE ALSO

DW_window_new

2.6. Constants/WindowGravityValues [Definitions]

[Top] [Constants] [Definitions]

NAME

WindowGravityValues

DESCRIPTION

The following constants are used in DW_window_set_gravity() to specify the edge that a window's position is relative to.

ATTRIBUTES

- DW_GRAV_CENTER - position of window in both horizontal and vertical
- DW_GRAV_CENTRE - position of window in both horizontal and vertical
- DW_GRAV_LEFT - window x position relative to left edge of screen
- DW_GRAV_RIGHT - window x position relative to right edge of screen
- DW_GRAV_TOP - window y position relative to top edge of screen
- DW_GRAV_BOTTOM - window y position relative to bottom edge of screen

SEE ALSO

DW_window_set_gravity

2.7. Constants/ContainerColumnFlags [Definitions]

[Top] [Constants] [Definitions]

NAME

ContainerColumnFlags

DESCRIPTION

The following is a list of the pre-defined flags that can be used when creating a container widget. When used in DW_container_setup() or DW_filesystem_setup(), one value from "Column Types", one value from "Alignment" and any value from "Separators" are logically "or"ed together using DW_or().

ATTRIBUTES

Column Types:

```
- DW_CFA_BITMAPORICON   - the column contains an image
- DW_CFA_STRING         - the column contains an unstructured string
- DW_CFA_ULONG          - the column contains an integer
- DW_CFA_TIME           - the column contains a time in the format Time('N')
- DW_CFA_DATE           - the column contains a date in the format Date('S')
- DW_CFA_STRINGANDICON  - the column contains a string and an image (not implemented)
```

Alignment:

```
- DW_CFA_CENTER         - the column is centered
- DW_CFA_CENTRE         - the column is centred
- DW_CFA_LEFT           - the column is left aligned
- DW_CFA_RIGHT          - the column is right aligned
```

Separators (not applicable on all platforms):

```
- DW_CFA_HORZSEPARATOR  - the column has a cell border on the top and bottom
- DW_CFA_SEPARATOR      - the column has a cell border on the left and right
```

SEE ALSO

DW_container_setup, DW_filesystem_setup

2.8. Constants/ContainerQueryFlags [Definitions]

[Top] [Constants] [Definitions]

NAME

ContainerQueryFlags

DESCRIPTION

The following is a list of the pre-defined flags that can be used when querying a container widget.

ATTRIBUTES

- DW_CRA_ALL - all rows are queried
- DW_CRA_SELECTED - only rows selected are queried
- DW_CRA_CURSORED - only the row(s) currently have focus; ie. highlighted

SEE ALSO

DW_container_query_start, DW_container_query_next

2.9. Constants/MLESearchFlags [Definitions]

[Top] [Constants] [Definitions]

NAME

MLESearchFlags

DESCRIPTION

The following is a list of the pre-defined flags that can be used in the SearchFlags argument in DW_mle_search().

ATTRIBUTES

- DW_MLE_CASESENSITIVE - search done respecting case

SEE ALSO

DW_mle_search

2.10. Constants/MLEEditableFlags [Definitions]

[Top] [Constants] [Definitions]

NAME

MLEEditableFlags

DESCRIPTION

The following is a list of the pre-defined flags that can be used in the EditableFlags argument in DW_mle_set_editable().

ATTRIBUTES

- DW_EDITABLE - MLE is editable
- DW_READONLY - MLE is not editable; ie readonly

SEE ALSO

DW_mle_set_editable

2.11. Constants/MLEWordWrapFlags [Definitions]

[Top] [Constants] [Definitions]

NAME

MLEWordWrapFlags

DESCRIPTION

The following is a list of the pre-defined flags that can be used in the WordWrapFlags argument in DW_mle_set_word_wrap().

ATTRIBUTES

- DW_WORD_WRAP - words are wrapped in the MLE
- DW_DONT_WORD_WRAP - no word wrapping occurs in the MLE

SEE ALSO

DW_mle_set_word_wrap

2.12. Constants/PointerTypes [Definitions]

[Top] [Constants] [Definitions]

NAME

PointerTypes

DESCRIPTION

The following is a list of the pre-defined mouse pointer types used in the Pointer argument in DW_window_set_pointer().

ATTRIBUTES

- DW_POINTER_DEFAULT - the default pointer shape
- DW_POINTER_CLOCK - a clock pointer
- DW_POINTER_ARROW - an arrow pointing to the top left
- DW_POINTER_QUESTION - a question mark pointer

SEE ALSO

DW_window_set_pointer

2.13. Constants/ButtonMasks [Definitions]

[Top] [Constants] [Definitions]

NAME

ButtonMasks

DESCRIPTION

The following is a list of the pre-defined mouse button mask types returned in the Button argument in a MotionNotify callback.

ATTRIBUTES

- DW_BUTTON1_MASK - button 1 is being pressed
- DW_BUTTON2_MASK - button 2 is being pressed
- DW_BUTTON3_MASK - button 3 is being pressed

SEE ALSO

MotionNotifyEventCallback

2.14. Constants/Colours [Definitions]

[Top] [Constants] [Definitions]

NAME

Colours

DESCRIPTION

The following is a list of the pre-defined colours known to RexxDW. These colour constants are used in functions that require pre-defined colours.

ATTRIBUTES

- DW_CLR_BLACK
- DW_CLR_DARKRED
- DW_CLR_DARKGREEN
- DW_CLR_BROWN
- DW_CLR_DARKBLUE
- DW_CLR_DARKPINK
- DW_CLR_DARKCYAN
- DW_CLR_PALEGRAY
- DW_CLR_DARKGRAY
- DW_CLR_RED
- DW_CLR_GREEN
- DW_CLR_YELLOW
- DW_CLR_BLUE
- DW_CLR_PINK
- DW_CLR_CYAN
- DW_CLR_WHITE
- DW_CLR_DEFAULT
- DW_CLR_TRANSPARENT (use on background)

SEE ALSO

DW_color_background_set, DW_color_foreground_set, DW_color_choose

2.15. Constants/WidgetOrientation [Definitions]

[Top] [Constants] [Definitions]

NAME

WidgetOrientation

DESCRIPTION

The following constants are used in widgets that require an orientation to be specified.

ATTRIBUTES

- DW_VERT - vertical orientation
- DW_HORZ - horizontal orientation

SEE ALSO

DW_box_new, DW_groupbox_new, DW_splitbar_new, DW_scrollbar_new

2.16. Constants/HTMLActions [Definitions]

[Top] [Constants] [Definitions]

NAME

HTMLActions

DESCRIPTION

The following constants are used with DW_html_action to take actions on an embedded HTML widget.

ATTRIBUTES

- DW_HTML_GOBACK - go back to the previously viewed page
- DW_HTML_GOFORWARD - go forward to the previously viewed page
- DW_HTML_GOHOME - go to HOME page
- DW_HTML_RELOAD - reload the current page
- DW_HTML_STOP - stop loading the current page

SEE ALSO

DW_html_action

2.17. Constants/NotebookTabLocation [Definitions]

[Top] [Constants] [Definitions]

NAME

NotebookTabLocation

DESCRIPTION

The following constants are used in the TabLocation argument in a call to DW_notebook_new() to specify whether notebook tabs are located at the top or bottom of the window.

ATTRIBUTES

- DW_TAB_TO_TOP - notebook tabs are displayed across the top
- DW_TAB_TO_BOTTOM - notebook tabs are displayed across the bottom

SEE ALSO

DW_notebook_new

2.18. Constants/NotebookPageLocation [Definitions]

[Top] [Constants] [Definitions]

NAME

NotebookPageLocation

DESCRIPTION

The following constants are used in the PageLocation argument in a call to DW_notebook_page_new() to specify if the page is created in front or behind other pages.

ATTRIBUTES

- DW_PAGE_TO_FRONT - page is created in front of other pages
- DW_PAGE_TO_BACK - page is created behind other pages

SEE ALSO

DW_notebook_page_new

2.19. Constants/MenuConstants [Definitions]

[Top] [Constants] [Definitions]

NAME

MenuConstants

DESCRIPTION

The following constants are used when appending a menu item to a menu.

ATTRIBUTES

- Title agument: - DW_MENU_SEPARATOR - the string to indicate the menu item is a separator
- Id agument: - DW_MENU_POPUP - indicates the item is part of a popup menu item - DW_MENU_AUTO - automatically assigns a menu id
- End agument: - DW_MENU_START - add the item at the start of the menu items - DW_MENU_END - add the item at the end of the menu items
- Check agument: - DW_MENU_CHECKABLE - the menu item is checkable - DW_MENU_NOT_CHECKABLE - the menu item is not checkable
- Submenu agument: - DW_MENU_NOMENU - indicates there is no submenu for this menu item

SEE ALSO

DW_menu_append_item

2.20. Constants/MenuItemConstants [Definitions]

[Top] [Constants] [Definitions]

NAME

MenuItemConstants

DESCRIPTION

The following constants are used when setting the state of a menu item.

ATTRIBUTES

- DW_MENU_CHECKED - sets the state of the checkable menu item to checked
- DW_MENU_UNCHECKED - sets the state of the checkable menu item to unchecked
- DW_MENU_ENABLED - enables the menu item
- DW_MENU_DISABLED - disables the menu item

SEE ALSO

DW_menu_item_set_state, DW_or

NOTES

Any of the above attributes can be ORed together.

2.21. Constants/BoxExpansionFlags [Definitions]

[Top] [Constants] [Definitions]

NAME

BoxExpansionFlags

DESCRIPTION

The following constants are used in the box packing functions to specify whether the box expand in a horizontal or vertical direction.

ATTRIBUTES

- DW_EXPAND_HORZ - expand the box horizontally
- DW_DONT_EXPAND_HORZ - don't expand the box horizontal
- DW_EXPAND_VERT - expand the box vertically
- DW_DONT_EXPAND_VERT - don't expand the box vertically

SEE ALSO

DW_box_pack_start, DW_box_pack_end, DW_box_pack_at_index

2.22. Constants/BoxSizeValues [Definitions]

[Top] [Constants] [Definitions]

NAME

BoxSizeValues

DESCRIPTION

The following constant is used in the box packing functions where a pixel size would normally be specified. This constant value indicates that Rexx/DW should automatically size the widget in the direction indicated.

ATTRIBUTES

- DW_SIZE_AUTO - size the box automatically in the indicated direction

SEE ALSO

DW_box_pack_start, DW_box_pack_end

2.23. Constants/GroupBoxFontFlags [Definitions]

[Top] [Constants] [Definitions]

NAME

GroupBoxFontFlags

DESCRIPTION

The following constants can be used when creating a groupbox to indicate whether the font used for displaying the title text is normal, bold and/or italic. Use DW_or to specify bold and italic. This flag is optional and defaults to DW_FONT_NORMAL.

ATTRIBUTES

- DW_FONT_NORMAL - use the default font
- DW_FONT_BOLD - use the default font with bold attribute
- DW_FONT_ITALIC - use the default font with italic attribute

SEE ALSO

DW_groupbox_new

2.24. Constants/ListboxSelection [Definitions]

[Top] [Constants] [Definitions]

NAME

ListboxSelection

DESCRIPTION

The following constants are used in the Selection argument to a call to DW_listbox_new().

ATTRIBUTES

- DW_LB_SINGLE_SELECTION - indicates that only 1 row in the listbox is selectable
- DW_LB_MULTIPLE_SELECTION - indicates that multiple rows in the listbox are selectable

SEE ALSO

DW_listbox_new

2.25. Constants/ListboxSelected [Definitions]

[Top] [Constants] [Definitions]

NAME

ListboxSelected

DESCRIPTION

The following is a list of the pre-defined flags that can be used when setting the State of a listbox or combobox.

ATTRIBUTES

- DW_LB_SELECTED - set the item to be selected
- DW_LB_UNSELECTED - set the item to NOT be selected

SEE ALSO

DW_listbox_select

2.26. Constants/ListboxReturn [Definitions]

[Top] [Constants] [Definitions]

NAME

ListboxReturn

DESCRIPTION

The following is the return value from DW_listbox_selected() or DW_listbox_selected_multi() if no item in the listbox has been selected.

ATTRIBUTES

- DW_LB_NONE - no listbox item selected

SEE ALSO

DW_listbox_selected(), DW_listbox_selected_multi()

2.27. Constants/WidgetChecked [Definitions]

[Top] [Constants] [Definitions]

NAME

WidgetChecked

DESCRIPTION

The following constants are used to indicate the state of a widget that can have a boolean state.

ATTRIBUTES

- DW_CHECKED - widget is "on" or set
- DW_UNCHECKED - widget is "off" or unset

SEE ALSO

DW_checkbox_set, DW_radiobutton_set

NOTES

For

2.28. Constants/ContainerClearFlags [Definitions]

[Top] [Constants] [Definitions]

NAME

ContainerClearFlags

DESCRIPTION

The following constants are used in the Redraw argument to a call to DW_container_clear().

ATTRIBUTES

- DW_REDRAW - redraw the cleared container
- DW_DONT_REDRAW - don't redraw the cleared container

SEE ALSO

DW_container_new

2.29. Constants/ContainerSelection [Definitions]

[Top] [Constants] [Definitions]

NAME

ContainerSelection

DESCRIPTION

The following constants are used in the Selection argument to a call to DW_container_new().

ATTRIBUTES

- DW_SINGLE_SELECTION - indicates that only 1 row in the container is selectable
- DW_MULTIPLE_SELECTION - indicates that multiple rows in the container are selectable

SEE ALSO

DW_container_new

2.30. Constants/DrawingFlags [Definitions]

[Top] [Constants] [Definitions]

NAME

DrawingFlags

DESCRIPTION

The following constants are used in the Flags argument to a call to DW_draw_rect(), DW_draw_arc(), or DW_draw_polygon.

ATTRIBUTES

- DW_DRAW_DEFAULT - draws an outline shape 1 pixel in width
- DW_DRAW_FILL - draws a filled shape
- DW_DRAW_FULL - draws a circle when DW_draw_arc() is called
- DW_DRAW_NOAA - turns off anti-aliasing
- DW_FILL - draws a filled shape (deprecated)
- DW_DONT_FILL - draws an outline shape 1 pixel in width (deprecated)

SEE ALSO

DW_draw_rect, DW_draw_arc, DW_draw_polygon

2.31. Constants/MessageboxFlags [Definitions]

[Top] [Constants] [Definitions]

NAME

MessageboxFlags

DESCRIPTION

The following constants are used in the Flags argument to a call to DW_messagebox(). Logically "or" one value from Buttons and one value from Icons

ATTRIBUTES

Buttons:

- DW_MB_OK
- DW_MB_OKCANCEL
- DW_MB_YESNO
- DW_MB_YESNOCANCEL

Icons:

- DW_MB_WARNING
- DW_MB_ERROR
- DW_MB_INFORMATION
- DW_MB_QUESTION

SEE ALSO

DW_messagebox

2.32. Constants/MessageboxResults [Definitions]

[Top] [Constants] [Definitions]

NAME

MessageboxResults

DESCRIPTION

The following constants are returned from a call to DW_messagebox().

ATTRIBUTES

- DW_MB_RETURN_OK
- DW_MB_RETURN_YES
- DW_MB_RETURN_NO
- DW_MB_RETURN_CANCEL

SEE ALSO

DW_messagebox

2.33. Constants/VirtualKeys [Definitions]

[Top] [Constants] [Definitions]

NAME

VirtualKeys

DESCRIPTION

The following is a list of the mnemonic key values known to RexxDW. These keys are returned in a KeyPressEvent Callback, in the VirtualKey argument.

ATTRIBUTES

- DW_VK_CANCEL
- DW_VK_BACK
- DW_VK_TAB
- DW_VK_CLEAR
- DW_VK_RETURN
- DW_VK_MENU
- DW_VK_PAUSE
- DW_VK_CAPITAL
- DW_VK_ESCAPE
- DW_VK_SPACE
- DW_VK_PRIOR
- DW_VK_NEXT
- DW_VK_END
- DW_VK_HOME
- DW_VK_LEFT
- DW_VK_UP
- DW_VK_RIGHT
- DW_VK_DOWN
- DW_VK_SELECT
- DW_VK_PRINT
- DW_VK_EXECUTE
- DW_VK_SNAPSHOT
- DW_VK_INSERT
- DW_VK_DELETE
- DW_VK_HELP
- DW_VK_LWIN
- DW_VK_RWIN
- DW_VK_NUMPAD0
- DW_VK_NUMPAD1
- DW_VK_NUMPAD2
- DW_VK_NUMPAD3
- DW_VK_NUMPAD4
- DW_VK_NUMPAD5
- DW_VK_NUMPAD6
- DW_VK_NUMPAD7

- DW_VK_NUMPAD8
- DW_VK_NUMPAD9
- DW_VK_MULTIPLY
- DW_VK_ADD
- DW_VK_SEPARATOR
- DW_VK_SUBTRACT
- DW_VK_DECIMAL
- DW_VK_DIVIDE
- DW_VK_F1
- DW_VK_F2
- DW_VK_F3
- DW_VK_F4
- DW_VK_F5
- DW_VK_F6
- DW_VK_F7
- DW_VK_F8
- DW_VK_F9
- DW_VK_F10
- DW_VK_F11
- DW_VK_F12
- DW_VK_F13
- DW_VK_F14
- DW_VK_F15
- DW_VK_F16
- DW_VK_F17
- DW_VK_F18
- DW_VK_F19
- DW_VK_F20
- DW_VK_F21
- DW_VK_F22
- DW_VK_F23
- DW_VK_F24
- DW_VK_NUMLOCK
- DW_VK_SCROLL
- DW_VK_LSHIFT
- DW_VK_RSHIFT
- DW_VK_LCONTROL
- DW_VK_RCONTROL
- DW_VK_LMENU
- DW_VK_RMENU

SEE ALSO

KeyPressEventCallback

2.34. Constants/KeyStates [Definitions]

[Top] [Constants] [Definitions]

NAME

KeyStates

DESCRIPTION

The following is a list of the mnemonic key state values known to Rexx/DW. These keys are returned in a KeyPressEvent Callback, in the KeyState argument.

ATTRIBUTES

- DW_KC_CTRL
- DW_KC_SHIFT
- DW_KC_ALT

SEE ALSO

KeyPressEventCallback

2.35. Constants/MouseButtons [Definitions]

[Top] [Constants] [Definitions]

NAME

MouseButtons

DESCRIPTION

The following is a list of the mnemonic mouse button values known to RexxDW. These buttons are returned in a ButtonPressEvent Callback, in the Button argument.

ATTRIBUTES

- DW_VK_LBUTTON
- DW_VK_RBUTTON
- DW_VK_MBUTTON

SEE ALSO

ButtonPressEventCallback, ButtonReleaseEventCallback

2.36. Constants/PercentValue [Definitions]

[Top] [Constants] [Definitions]

NAME

PercentValue

DESCRIPTION

The following constant is used as the value argument to DW_percent_set_pos() for displaying a moving bar which is not relative to a fixed value.

ATTRIBUTES

- DW_PERCENT_INDETERMINATE

SEE ALSO

DW_percent_set_pos()

3. RexxDW/Fontnames [Modules]

[Top] [Modules]

NAME

Fontnames

DESCRIPTION

Fonts are specified in Rexx/DW in Dynamic Windows font format on all platforms. This format is:

```
fontsize.fontname [bold] [italic]
```

For example to set the font of a widget to 14 point Courier Bold:

```
Call dw_window_set_font win, '14.Courier Bold'
```

NOTES

While the format of the font specification is the same on all platforms, the fontnames available on each platform will be different.

On GTK+ 2.x+ it is also possible to specify fonts in Pango format:

```
fontname [bold] [italic] fontsize
```

when setting a font. However, dw_window_get_font() will always return the Dynamic Windows font format.

SEE ALSO

DW_window_set_font, DW_window_get_font

4. RexxDW/Packing [Modules]

[Top] [Modules]

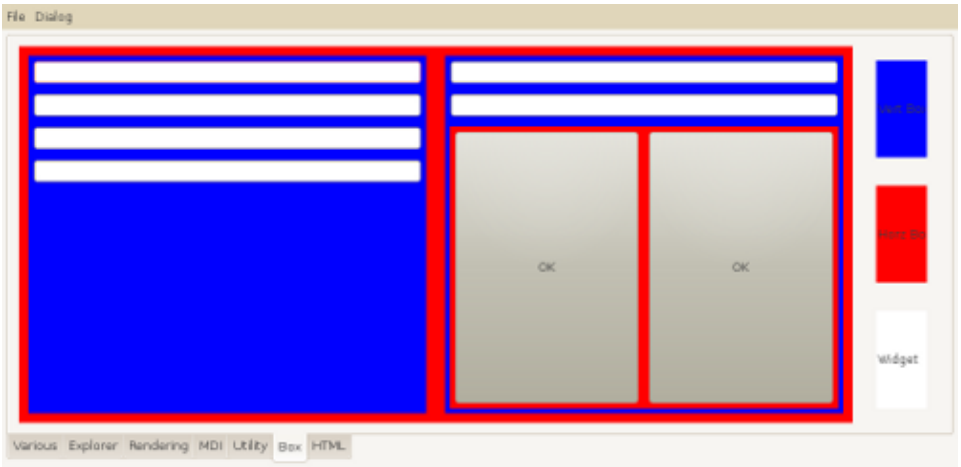
NAME

Packing

DESCRIPTION

One of the most important aspects of laying out widgets in a Rexx/DW program is how packing works. This section explains the Rexx/DW packing mechanism.

NOTES



The above ?????? TODO - horz/vert boxes, pack empty spaces, expand/dont_expand

5. RexxDW/Callbacks [Modules]

[Top] [Modules]

NAME

Callbacks

DESCRIPTION

Rexx/DW is an event driven application. What this means is that internally various events occur, such as a key press, a window becoming exposed. Rexx/DW allows you to call an internal Rexx procedure when various events occur.

Each widget you create has certain valid events associated with it. As part of your code you associate a particular event with a widget. This is done by calling DW_signal_connect(). This associates an event on a particular widget with an internal Rexx procedure, so that when the particular event occurs on that widget your procedure is executed and certain parameters are passed to it. Details on what parameters are passed is detailed below depending on the type of event.

The internal Rexx procedure is called in the context of where DW_main() or DW_main_iteration() is called. This is important when considering what Rexx variables may or may not be in scope. It is probably better to always use PROCEDURE EXPOSE (myglobals) on procedures so you know that any variables you use within your procedure will not affect an others outside without you explicitly listing them.

Once you have handled an event in your code, you can return 1 to inform Dynamic Windows that all processing has been handled for the particular event. Dynamic Windows will not execute its default event handling code. If your event handler returns 0, then Dynamic Windows will execute its default event handling code.

ATTRIBUTES

The values of the supported events are:

- REXXDW_CONFIGURE_EVENT_CB
- REXXDW_KEY_PRESS_EVENT_CB
- REXXDW_BUTTON_PRESS_EVENT_CB
- REXXDW_BUTTON_RELEASE_EVENT_CB
- REXXDW_MOTION_NOTIFY_EVENT_CB
- REXXDW_DELETE_EVENT_CB
- REXXDW_EXPOSE_EVENT_CB
- REXXDW_CLICKED_EVENT_CB
- REXXDW_ITEM_ENTER_EVENT_CB
- REXXDW_ITEM_CONTEXT_EVENT_CB
- REXXDW_LIST_SELECT_EVENT_CB
- REXXDW_ITEM_SELECT_EVENT_CB
- REXXDW_SET_FOCUS_EVENT_CB
- REXXDW_VALUE_CHANGED_EVENT_CB
- REXXDW_SWITCH_PAGE_EVENT_CB
- REXXDW_COLUMN_CLICK_EVENT_CB

NOTES

With Rexx interpreters that don't provide RexxCallBack() it is not possible to inform Dynamic Windows to execute the default callback functionality after your event handler has executed its code, because the return value is returned to the Rexx program; via DW_main_iteration(), not to Dynamic Windows.

5.1. Callbacks/ConfigureEventCallback [Generics]

[Top] [Callbacks] [Generics]

NAME

ConfigureEventCallback

USAGE

rcode = **ConfigureEventCallback**(Window, Width, Height [,UserData[,...]])

DESCRIPTION

Called when a DW_CONFIGURE_EVENT signal fires on a window. This event usually occurs when a window's size is changed.

ARGUMENTS

- Window - value of the window handle that has been resized
- Width - the new width of Window in pixels
- Height - the new height of Window in pixels
- UserData - following optional values that the user supplied when the signal event was connected

RETURN VALUE

Return 1 to ensure that the default callback is not called. Return 0 to enable the default callback to be called.

SOURCE

```
...
Call dw_signal_connect window, !REXXDW.!DW_CONFIGURE_EVENT, 'configure_cb', ,
   'fred', 'mary'
...
configure_cb: Procedure Expose !REXXDW.
Parse Arg win, width, height, var1, var2
Say 'New width:' width 'height:' height 'for Window:' win 'UserVar1:' var1 ,
   'UserVar2:' var2
Return 1
```

5.2. Callbacks/KeyPressEventCallback [Generics]

[Top] [Callbacks] [Generics]

NAME

KeyPressEventCallback

USAGE

rcode = **KeyPressEventCallback**(Window, ASCIIKey, VirtualKey, KeyState [,UserData[,...]])

DESCRIPTION

Called when a DW_KEY_PRESS_EVENT signal fires on a window. Occurs when the user presses a key.

ARGUMENTS

- Window - value of the window handle in which the key was pressed
- ASCIIKey - the Hex representation of the key pressed. If not as ASCII key then the empty string is returned and VirtualKey should be used
- VirtualKey - the mnemonic key value of the key pressed
- KeyState - indicates whether SHIFT, CTRL or ALT modifiers are active
- UserData - following optional values that the user supplied when the signal event was connected

RETURN VALUE

Return 1 to ensure that the default callback is not called. Return 0 to enable the default callback to be called.

SEE ALSO

VirtualKeys, KeyStates

SOURCE

```
...
Call dw_signal_connect window, !REXXDW.!DW_KEY_PRESS_EVENT, 'keypress_cb', ,
   'fred', 'mary'
...
keypress_cb: Procedure Expose !REXXDW.
Parse Arg win, ascii, virtual, state, var1, var2
Say 'ASCII Key:' ascii 'Virtual Key:' virtual 'State:' state 'UserVar1:' ,
   var1 'UserVar2:' var2
If virtual = !REXXDW.!DW_VK_F1 Then Return 1
Return 1
```

5.3. Callbacks/ButtonPressEventCallback [Generics]

[Top] [Callbacks] [Generics]

NAME

ButtonPressEventCallback

USAGE

rcode = **ButtonPressEventCallback**(Window, X, Y, Button [,UserData[,...]])

DESCRIPTION

Called when a DW_BUTTON_PRESS_EVENT signal fires on a window. This occurs when the user clicks a button with the mouse.

ARGUMENTS

- Window - value of the window handle in which the mouse button was pressed
- X - the X coordinate where the mouse button was pressed, relative to the top left corner of the Window
- Y - the Y coordinate where the mouse button was pressed, relative to the top left corner of the Window
- Button - which mouse button was pressed
- UserData - following optional values that the user supplied when the signal event was connected

RETURN VALUE

Return 1 to ensure that the default callback is not called. Return 0 to enable the default callback to be called.

SEE ALSO

MouseButtons

SOURCE

```
...
Call dw_signal_connect window, !REXXDW.!DW_BUTTON_PRESS_EVENT, ,
   'buttonpress_cb', 'fred'
...
buttonpress_cb:
Parse Arg win, x, y, button, data
Say 'Button:' button 'pressed at:' x'/'y 'in Window:' win
Return 1
```

5.4. Callbacks/ButtonReleaseEventCallback [Generics]

[Top] [Callbacks] [Generics]

NAME

ButtonReleaseEventCallback

USAGE

rcode = **ButtonReleaseEventCallback**(Window, X, Y, Button [,UserData[,...]])

DESCRIPTION

Called when a DW_BUTTON_RELEASE_EVENT signal fires on a window. This occurs when the user releases a button on the mouse.

ARGUMENTS

- Window - value of the window handle in which the mouse button was released
- X - the X coordinate where the mouse button was released, relative to the top left corner of the Window
- Y - the Y coordinate where the mouse button was released, relative to the top left corner of the Window
- Button - which mouse button was released
- UserData - following optional values that the user supplied when the signal event was connected

RETURN VALUE

Return 1 to ensure that the default callback is not called. Return 0 to enable the default callback to be called.

SEE ALSO

MouseButtons

SOURCE

```
...
Call dw_signal_connect window, !REXXDW.!DW_BUTTON_RELEASE_EVENT, ,
   'buttonrelease_cb', 'fred'
...
buttonreleases_cb:
Parse Arg win, x, y, button, data
Say 'Button:' button 'released at:' x'/'y 'in Window:' win
Return 1
```

5.5. Callbacks/MotionNotifyEventCallback [Generics]

[Top] [Callbacks] [Generics]

NAME

MotionNotifyEventCallback

USAGE

rcode = **MotionNotifyEventCallback**(Window, X, Y, Button [,UserData[,...]])

DESCRIPTION

Called when a DW_MOTION_NOTIFY_EVENT signal fires on a window. This occurs when the user moves the mouse.

ARGUMENTS

- Window - value of the window handle in which the mouse is being moved
- X - the X coordinate where the mouse cursor is currently displayed, relative to the top left corner of the Window
- Y - the Y coordinate where the mouse cursor is currently displayed, relative to the top left corner of the Window
- Button - Indicates which button(s) are being pressed during the mouse
- UserData - following optional values that the user supplied when the signal event was connected

RETURN VALUE

Return 1 to ensure that the default callback is not called. Return 0 to enable the default callback to be called.

SEE ALSO

ButtonMasks, ButtonPressEventCallback, ButtonReleaseEventCallback

SOURCE

```
 ...
Call dw_signal_connect window, !REXXDW.!DW_MOTION_NOTIFY_EVENT, ,
   'motionnotify_cb', 'fred'
...
motionnotify_cb:
Parse Arg win, x, y, state, data
Say 'Mouse movied to' x'/'y 'in Window:' win 'with state:' state
Return 1
```

5.6. Callbacks/DeleteEventCallback [Generics]

[Top] [Callbacks] [Generics]

NAME

DeleteEventCallback

USAGE

rcode = **DeleteEventCallback**(Window [,UserData[,...]])

DESCRIPTION

Called when a DW_DELETE_EVENT signal fires on a window. This occurs when the user closes the window by clicking the close icon on the window title bar, or when dw_window_destroy() ???? is called on

the window.

ARGUMENTS

- Window - value of the window handle in which the mouse button was released
- UserData - following optional values that the user supplied when the signal event was connected

RETURN VALUE

Return 1 to ensure that the default callback is not called. Return 0 to enable the default callback to be called.

SEE ALSO

DW_window_destroy

SOURCE

```
...
Call dw_signal_connect window, !REXXDW.!DW_DELETE_EVENT, 'delete_cb', ,
   'fred'
...
delete_cb:
Parse Arg win, data
Say 'Window:' win 'deleted'
Return 1
```

5.7. Callbacks/ExposeEventCallback [Generics]

[Top] [Callbacks] [Generics]

NAME

ExposeEventCallback

USAGE

rcode = **ExposeEventCallback**(Window, X, Y, Width, Height [,UserData[,...]])

DESCRIPTION

Called when a DW_EXPOSE_EVENT signal fires on a window. This occurs when the window is exposed.

ARGUMENTS

- Window - value of the window handle which is being exposed
- X - the top left X coordinate of the exposed area
- Y - the top left Y coordinate of the exposed area
- Width - the width of the exposed area
- Height - the height of the exposed area
- UserData - following optional values that the user supplied when the signal event was connected

RETURN VALUE

Return 1 to ensure that the default callback is not called. Return 0 to enable the default callback to be called.

SOURCE

```
...
Call dw_signal_connect window, !REXXDW.!DW_EXPOSE_EVENT, 'expose_cb', ,
   'fred'
...
expose_cb:
Parse Arg win, x, y, width, height, data
Say 'Window:' win 'exposed at' x'/'y 'size:' width'/'height
Return 1
```

5.8. Callbacks/ClickedEventCallback [Generics]

[Top] [Callbacks] [Generics]

NAME

ClickedEventCallback

USAGE

rcode = **ClickedEventCallback**(Window [,UserData[,...]])

DESCRIPTION

Called when a DW_CLICKED_EVENT signal fires on a window. This occurs when the user clicks in the window; usually a button window.

ARGUMENTS

- Window - value of the window handle that was clicked
- UserData - following optional values that the user supplied when the signal event was connected

RETURN VALUE

Return 1 to ensure that the default callback is not called. Return 0 to enable the default callback to be called.

SOURCE

```
...
Call dw_signal_connect window, !REXXDW.!DW_CLICKED_EVENT, 'clicked_cb', ,
   'fred'
...
clicked_cb:
Parse Arg win, data
Say 'Window:' win 'clicked'
Return 1
```

5.9. Callbacks/ItemEnterEventCallback [Generics]

[Top] [Callbacks] [Generics]

NAME

ItemEnterEventCallback

USAGE

rcode = **ItemEnterEventCallback**(Window, Text [,UserData[,...]])

DESCRIPTION

Called when a DW_ITEM_ENTER_EVENT signal fires on a window. This occurs when the user double-clicks or presses ENTER key in a container window.

ARGUMENTS

- Window - value of the window handle that was clicked
- Text - the string set by a call to DW_container_set_row_title()
- UserData - following optional values that the user supplied when the signal event was connected

RETURN VALUE

Return 1 to ensure that the default callback is not called. Return 0 to enable the default callback to be called.

SEE ALSO

DW_container_set_row_title

SOURCE

```
...
Call dw_signal_connect window, !REXXDW.!DW_ITEM_ENTER_EVENT, 'itementer_cb', ,
   'fred'
...
itementer_cb:
Parse Arg win, text, data
Say 'Item text' text 'entered in Window:' win 'userdata:' data
Return 1
```

5.10. Callbacks/ItemContextEventCallback [Generics]

[Top] [Callbacks] [Generics]

NAME

ItemContextEventCallback

USAGE

rcode = **ItemContextEventCallback**(Window, Text, X, Y, ItemData [,UserData[,...]])

DESCRIPTION

Called when a DW_ITEM_CONTEXT_EVENT signal fires on a window. This occurs when the user clicks the right mouse button in a container or tree window.

ARGUMENTS

- Window - value of the window handle that was clicked
- Text - the string set by a call to DW_container_set_row_title() or the third argument ???? to DW_tree_item_change()
- X - the X coordinate where the mouse button was pressed, relative to the top left corner of the Window
- Y - the Y coordinate where the mouse button was pressed, relative to the top left corner of the Window
- ItemData - ????
- UserData - following optional values that the user supplied when the signal event was connected

RETURN VALUE

Return 1 to ensure that the default callback is not called. Return 0 to enable the default callback to be called.

SEE ALSO

DW_container_set_row_title, DW_tree_item_change

NOTES

When connecting this signal to a Tree widget, DW_signal_connect() MUST be called BEFORE the Tree is populated. When connecting this signal to a Container widget, DW_signal_connect() MUST be called AFTER the Container is populated.

SOURCE

```
...
Call dw_signal_connect window, !REXXDW.!DW_ITEM_CONTEXT_EVENT, ,
   'itemcontext_cb', 'fred'
...
itemcontext_cb:
Parse Arg win, text, x, y, itemdata, userdata
Say 'Item text' text 'entered in Window:' win 'at' x'/'y 'userdata:' userdata
Return 1
```

5.11. Callbacks/ListSelectEventCallback [Generics]

[Top] [Callbacks] [Generics]

NAME

ListSelectEventCallback

USAGE

rcode = **ListSelectEventCallback**(Window, Item [,UserData[,...]])

DESCRIPTION

Called when a DW_LIST_SELECT_EVENT signal fires on a window. This occurs when the user clicks an entry in a listbox or combobox; ie an item in a list is selected.

ARGUMENTS

- Window - value of the window handle that was clicked
- Item - the index of the item in the list that is selected. This index is 0-based.
- UserData - following optional values that the user supplied when the signal event was connected

RETURN VALUE

Return 1 to ensure that the default callback is not called. Return 0 to enable the default callback to be called.

SOURCE

```
...
Call dw_signal_connect window, !REXXDW.!DW_LIST_SELECT_EVENT, ,
   'listselect_cb', 'fred'
...
listselect_cb:
Parse Arg win, item, userdata
Say 'Item number' item 'selected in Window:' win 'userdata:' userdata
Return 1
```

5.12. Callbacks/ItemSelectEventCallback [Generics]

[Top] [Callbacks] [Generics]

NAME

ItemSelectEventCallback

USAGE

rcode = **ItemSelectEventCallback**(Window, Item, Text, Itemdata [,UserData[,...]])

DESCRIPTION

Called when a DW_ITEM_SELECT_EVENT signal fires on a window. This occurs when the user clicks the left button on an item in a container or tree window. selected.

ARGUMENTS

- Window - value of the window handle that was clicked
- Item - the index of the item in the list that is selected. This index is 0-based.
- Text - the string set by a call to DW_container_set_row_title() or the second argument to DW_tree_insert()
- ItemData - the string set by a the fifth argument to a call to DW_tree_insert()
- UserData - following optional values that the user supplied when the signal event was connected

RETURN VALUE

Return 1 to ensure that the default callback is not called. Return 0 to enable the default callback to be called.

NOTES

When connecting this signal to a Tree widget, DW_signal_connect() MUST be called BEFORE the Tree is populated otherwise the signal will never fire. When connecting this signal to a Container widget, DW_signal_connect() MUST be called AFTER the Container is populated.

SOURCE

```
...
Call dw_signal_connect window, !REXXDW.!DW_ITEM_SELECT_EVENT, ,
   'itemselect_cb', 'fred'
...
itemselect_cb:
Parse Arg win, item, text, userdata
Say 'Item number' item 'selected in Window:' win 'text is:' text ,
   'userdata:' userdata
Return 1
```

5.13. Callbacks/SetFocusEventCallback [Generics]

[Top] [Callbacks] [Generics]

NAME

SetFocusEventCallback

USAGE

rcode = **SetFocusEventCallback**(Window [,UserData[,...]])

DESCRIPTION

Called when a DW_SET_FOCUS_EVENT signal fires on a window. This occurs when the user gives focus to a window, either by clicking the left button in the window or by tabbing to the window. selected.

ARGUMENTS

- Window - value of the window handle that was clicked
- UserData - following optional values that the user supplied when the signal event was connected

RETURN VALUE

Return 1 to ensure that the default callback is not called. Return 0 to enable the default callback to be called.

SOURCE

```
...
Call dw_signal_connect window, !REXXDW.!DW_SET_FOCUS_EVENT, 'setfocus_cb', ,
   'fred'
...
setfocus_cb:
Parse Arg win, userdata
Say 'Window:' win 'now has focus userdata:' userdata
Return 1
```

5.14. Callbacks/ValueChangedEventCallback [Generics]

[Top] [Callbacks] [Generics]

NAME

ValueChangedEventCallback

USAGE

rcode = **ValueChangedEventCallback**(Window, Value [,UserData[,...]])

DESCRIPTION

Called when a DW_VALUE_CHANGED_EVENT signal fires on a window. This occurs when the user adjusts a scrollbar, or a slider control.

ARGUMENTS

- Window - value of the window handle that was clicked
- Value - the new value of the top of the scrollbar thumb in units defined by DW_scrollbar_set_range()
- UserData - following optional values that the user supplied when the signal event was connected

RETURN VALUE

Return 1 to ensure that the default callback is not called. Return 0 to enable the default callback to be called.

SEE ALSO

DW_scrollbar_set_range, DW_slider_set_pos

SOURCE

```
...
Call dw_signal_connect window, !REXXDW.!DW_VALUE_CHANGED_EVENT, ,
   'valuechanged_cb', 'fred'
...
valuechanged_cb:
Parse Arg win, val, userdata
Say 'The top of the scrollbar thumb is now' val 'in Window:' win ,
   'userdata:' userdata
Return 1
```

5.15. Callbacks/SwitchPageEventCallback [Generics]

[Top] [Callbacks] [Generics]

NAME

SwitchPageEventCallback

USAGE

rcode = **SwitchPageEventCallback**(Window, Page [,UserData[,...]])

DESCRIPTION

Called when a DW_SWITCH_PAGE_EVENT signal fires on a window. This occurs when the user clicks on a tab in a notebook.

ARGUMENTS

- Window - value of the window handle that was clicked
- Page - the index of the page of the notebook selected. Page numbers are 0 based.
- UserData - following optional values that the user supplied when the signal event was connected

RETURN VALUE

Return 1 to ensure that the default callback is not called. Return 0 to enable the default callback to be called.

SOURCE

```
...
Call dw_signal_connect window, !REXXDW.!DW_SWITCH_PAGE_EVENT, ,
   'switchpage_cb', 'fred'
...
switchpage_cb:
Parse Arg win, page, userdata
```

```
Say 'Page' page 'of Window:' win 'selected. userdata:' userdata
Return 1
```

5.16. Callbacks/ColumnClickEventCallback [Generics]

[Top] [Callbacks] [Generics]

NAME

ColumnClickEventCallback

USAGE

rcode = **ColumnClickEventCallback**(Window, Column [,UserData[,...]])

DESCRIPTION

Called when a DW_COLUMN_CLICK_EVENT signal fires on a window. This occurs when the user clicks on a column heading of a container window. selected.

ARGUMENTS

- Window - value of the window handle that was clicked
- Column - the index of the column of the container selected. Column numbers are 0 based.
- UserData - following optional values that the user supplied when the signal event was connected

RETURN VALUE

Return 1 to ensure that the default callback is not called. Return 0 to enable the default callback to be called.

SOURCE

```
...
Call dw_signal_connect window, !REXXDW.!DW_COLUMN_CLICK_EVENT, ,
   'columnclick_cb', 'fred'
...
columnclick_cb:
Parse Arg win, column, userdata
Say 'Column' column 'of Window:' win 'selected. userdata:' userdata
Return 1
```

5.17. Callbacks/TimerEventCallback [Generics]

[Top] [Callbacks] [Generics]

NAME

TimerEventCallback

USAGE

rcode = **TimerEventCallback**([UserData[,...]])

DESCRIPTION

Called when a generic timer goes off. This occurs after the period specified by DW_timer_connect(). selected.

ARGUMENTS

- UserData - optional values that the user supplied when the signal event was connected

RETURN VALUE

Return 1 to ensure that the timer is rearmed. Return 0 to ensure that the timer is not rearmed.

NOTES

The return code from the function called as a result of the timer firing determines whether the timer continues to fire after the specified period.

SEE ALSO

DW_timer_connect

SOURCE

```
...
Call dw_timer_connect window, 5000, 'timer_cb', 'fred'
...
timer_cb:
Parse Arg userdata
Say 'Our timer went off after 5 seconds' 'userdata:' userdata
Return 1
```

6. RexxDW/Functions [Modules]

[Top] [Modules]

DESCRIPTION

The following external functions comprise Rexx/DW.

6.1. Functions/Widgets [Modules]

[Top] [Functions] [Modules]

DESCRIPTION

Widgets are the fundamental display items in RexxDW. RexxDW supports a range of widgets to enable a rich GUI interface for your Rexx programs.

NOTES

RexxDW supports all common widget types as follows:

```
box
button
checkbox/radiobutton
container/filesystem
dialog
rendering
entryfield
listbox/combobox
menu
multiline edit
notebook
percent
scrollbar
slider
spinbutton
splitbar
text
tree
window
```

6.1.1. Widgets/Box [Modules]

[Top] [Widgets] [Modules]

DESCRIPTION

Boxes are used to layout the GUI display. Widgets are packed into a box to position that widget. A box can be horizontal or vertical, which determines the arrangement of the widgets packed into the box. So to arrange

two widgets side-by-side, you would create a horizontal box, your two widgets, and pack those two widgets into the box and they appear beside each other.

6.1.1.1. Box/DW_box_new [Functions]

[Top] [Box] [Functions]

NAME

DW_box_new

SYNOPSIS

win = dw_box_new(Orientation)

FUNCTION

Creates a new generic box window. A box is the basic widget for arranging the location and position of other widgets.

ARGUMENTS

- Orientation- !REXXDW.!DW_VERT to indicate a vertical box !REXXDW.!DW_HORZ to indicate a horizontal box

RESULT

A window identifier.

SEE ALSO

DW_groupbox_new, WidgetOrientation

NOTES

The orientation of a box determines the orientation of the widgets that are packed into this box. Creating a horizontal box and packing two buttons into it will result in the two buttons being placed side by side.

SOURCE

```
...
box = dw_box_new( !REXXDW.!DW_VERT )
Call dw_box_pack_start win, box, 0, 0, !REXXDW.!DW_EXPAND_HORZ, ,
   !REXXDW.!DW_EXPAND_VERT, 0
```

6.1.1.2. Box/DW_scrollbox_new [Generics]

[Top] [Box] [Generics]

NAME

DW_scrollbox_new

SYNOPSIS

win = dw_scrollbox_new(Orientation)

FUNCTION

Creates a new generic box window with scrollbars. A box is the basic widget for arranging the location and position of other widgets.

ARGUMENTS

- Orientation- !REXXDW.!DW_VERT to indicate a vertical box !REXXDW.!DW_HORZ to indicate a horizontal box

RESULT

A window identifier.

SEE ALSO

DW_box_new, DW_groupbox_new, WidgetOrientation

NOTES

The orientation of a box determines the orientation of the widgets that are packed into this box. Creating a horizontal box and packing two buttons into it will result in the two buttons being placed side by side.

SOURCE

```
...
box = dw_scrollbox_new( !REXXDW.!DW_VERT )
Call dw_box_pack_start win, box, 0, 0, !REXXDW.!DW_EXPAND_HORZ, ,
   !REXXDW.!DW_EXPAND_VERT, 0
```

6.1.1.3. Box/DW_groupbox_new [Functions]

[Top] [Box] [Functions]

NAME

DW_groupbox_new

SYNOPSIS

win = dw_groupbox_new(Orientation, Title[, FontFlag])

FUNCTION

Creates a new generic groupbox window. A box is the basic widget for arranging the location and position of other widgets. The difference between a box and a groupbox is that a groupbox contains a border and a Title in the top left corner.

ARGUMENTS

- Orientation- !REXXDW.!DW_VERT to indicate a vertical box !REXXDW.!DW_HORZ to indicate a horizontal box
- Title - the text to display as the box title
- FontFlag - Optional. Specify !REXXDW.!DW_FONT_BOLD or !REXXDW.!DW_FONT_ITALIC to set the font attributes of Title. If tou require bold and italic or them together with DW_or. Specify !REXXDW.!DW_FONT_NORMAL or don't specify a value to use the normal, default font.

RESULT

A window identifier.

SEE ALSO

DW_box_new, WidgetOrientation

NOTES

The orientation of a box determines the orientation of the widgets that are packed into this box. Creating a horizontal box and packing two buttons into it will result in the two buttons being placed side by side.

SOURCE

```
...
box = dw_groupbox_new( !REXXDW.!DW_VERT, 'Editor', dw_or( !REXXDW.!DW_FONT_BOLD, !REXXDW.!DW_FONT
Call dw_box_pack_start win, box, 0, 0, !REXXDW.!DW_EXPAND_HORZ, ,
   !REXXDW.!DW_EXPAND_VERT, 0
```

6.1.1.4. Box/DW_box_pack_start [Functions]

[Top] [Box] [Functions]

NAME

DW_box_pack_start

SYNOPSIS

dw_box_pack_start(ParentBox, BoxToPack, BoxWidth, BoxHeight, HorzExpand, VertExpand, Padding)

FUNCTION

Packs the BoxToPack at the start of the ParentBox; ie at the left of a horizontal ParentBox, or the top of a vertical ParentBox. The height and width of the box can be specified as is the ability to allow the box to expand horizontally or vertically.

ARGUMENTS

- ParentBox - the box into which the BoxToPack is packed
- BoxToPack - the box that requires packing
- BoxWidth - the initial width of the box in pixels, or !REXXDW.!DW_SIZE_AUTO
- BoxHeight - the initial height of the box in pixels, or !REXXDW.!DW_SIZE_AUTO
- HorzExpand - indicate if the box can be expanded horizontally. Specify either !REXXDW.!DW_EXPAND_HORZ or !REXXDW.!DW_DONT_EXPAND_HORZ
- VertExpand - indicate if the box can be expanded vertically. Specify either !REXXDW.!DW_EXPAND_VERT or !REXXDW.!DW_DONT_EXPAND_VERT
- Padding - The number of pixels of padding to add around all sides of BoxToPack

RESULT

No return result.

SEE ALSO

DW_box_pack_end, DW_Box_pack_at_index, BoxSizeValues

NOTES

The BoxToPack argument can be 0, which will result in empty space being packed.
!REXXDW.!DW_SIZE_AUTO is used to enable the dwindows packing algorith to size the box automatically

SOURCE

```
...
button = dw_bitmapbutton_new_from_file( 'Quit', 0, 'quit' )
Call dw_box_pack_start win, button, 0, 32, !REXXDW.!DW_EXPAND_HORZ, ,
   !REXXDW.!DW_DONT_EXPAND_VERT, 0
```

6.1.1.5. Box/DW_box_pack_end [Functions]

[Top] [Box] [Functions]

NAME

DW_box_pack_end

SYNOPSIS

dw_box_pack_end(ParentBox, BoxToPack, BoxWidth, BoxHeight, HorzExpand, VertExpand, Padding)

FUNCTION

Packs the BoxToPack at the end of the ParentBox; ie at the right of a horizontal ParentBox, or the bottom of a vertical ParentBox. The height and width of the box can be specified as is the ability to allow the box to expand horizontally or vertically.

ARGUMENTS

- ParentBox - the box into which the BoxToPack is packed
- BoxToPack - the box that requires packing
- BoxWidth - the initial width of the box in pixels, or !REXXDW.!DW_SIZE_AUTO
- BoxHeight - the initial height of the box in pixels, or !REXXDW.!DW_SIZE_AUTO
- HorzExpand - indicate if the box can be expanded horizontally. Specify either !REXXDW.!DW_EXPAND_HORZ or !REXXDW.!DW_DONT_EXPAND_HORZ
- VertExpand - indicate if the box can be expanded vertically. Specify either !REXXDW.!DW_EXPAND_VERT or !REXXDW.!DW_DONT_EXPAND_VERT
- Padding - The number of pixels of padding to add around all sides of BoxToPack

RESULT

No return result.

SEE ALSO

DW_box_pack_start, DW_box_pack_at_index, BoxSizeValues

NOTES

The BoxToPack argument can be 0, which will result in empty space being packed.
!REXXDW.!DW_SIZE_AUTO is used to enable the dwindows packing algorith to size the box automatically

SOURCE

```
...
button = dw_bitmapbutton_new_from_file( 'Quit', 0, 'quit' )
Call dw_box_pack_end win, button, 0, 32, !REXXDW.!DW_EXPAND_HORZ, ,
   !REXXDW.!DW_DONT_EXPAND_VERT, 0
```

6.1.1.6. Box/DW_box_pack_at_index [Functions]

[Top] [Box] [Functions]

NAME

DW_box_pack_at_index

SYNOPSIS

dw_box_pack_at_index(ParentBox, BoxToPack, Index, BoxWidth, BoxHeight, HorzExpand, VertExpand, Padding)

FUNCTION

Packs the BoxToPack after the ?????????at the end of the ParentBox The height and width of the box can be specified as is the ability to allow the box to expand horizontally or vertically.

ARGUMENTS

- ParentBox - the box into which the BoxToPack is packed
- BoxToPack - the box that requires packing
- Index - ????
- BoxWidth - the initial width of the box in pixels, or !REXXDW.!DW_SIZE_AUTO
- BoxHeight - the initial height of the box in pixels, or !REXXDW.!DW_SIZE_AUTO
- HorzExpand - indicate if the box can be expanded horizontally. Specify either !REXXDW.!DW_EXPAND_HORZ or !REXXDW.!DW_DONT_EXPAND_HORZ
- VertExpand - indicate if the box can be expanded vertically. Specify either !REXXDW.!DW_EXPAND_VERT or !REXXDW.!DW_DONT_EXPAND_VERT
- Padding - The number of pixels of padding to add around all sides of BoxToPack

RESULT

No return result.

SEE ALSO

DW_box_pack_start, DW_box_pack_end, BoxSizeValues

NOTES

The BoxToPack argument can be 0, which will result in empty space being packed.
!REXXDW.!DW_SIZE_AUTO is used to enable the dwindows packing algorith to size the box automatically

SOURCE

```
...
button = dw_bitmapbutton_new_from_file( 'Quit', 0, 'quit' )
Call dw_box_pack_at_index win, button, 2, 0, 32, !REXXDW.!DW_EXPAND_HORZ, ,
   !REXXDW.!DW_DONT_EXPAND_VERT, 0
```

6.1.1.7. Box/DW_box_unpack [Functions]

[Top] [Box] [Functions]

NAME

DW_box_unpack

SYNOPSIS

win = dw_box_unpack(BoxToUnpack)

FUNCTION

Unpacks the BoxToUnpack from the current parent box.

ARGUMENTS

- BoxToUnpack - the box that is to be unpacked

RESULT

The window identifier of the box to be unpacked.

SEE ALSO

DW_box_unpack_at_index, DW_window_reparent

NOTES

The window identifier returned can be used to pack into another box.

SOURCE

```
...
box = dw_box_new( !REXXDW.!DW_VERT )
Call dw_box_pack_start win, box, 0, 0, !REXXDW.!DW_EXPAND_HORZ, ,
   !REXXDW.!DW_EXPAND_VERT, 0
...
win = dw_box_unpack( box )
```

6.1.1.8. Box/DW_box_unpack_at_index [Functions]

[Top] [Box] [Functions]

NAME

DW_box_unpack_at_index

SYNOPSIS

win = dw_box_unpack_at_index(BoxToUnpack, Index)

FUNCTION

Unpacks the BoxToUnpack ?????????

ARGUMENTS

- BoxToUnpack - the box that is to be unpacked

RESULT

The window identifier of the box to be unpacked.

SEE ALSO

DW_box_unpack, DW_window_reparent

NOTES

The window identifier returned can be used to pack into another box.

SOURCE

```
...
box = dw_box_new( !REXXDW.!DW_VERT )
Call dw_box_pack_start win, box, 0, 0, !REXXDW.!DW_EXPAND_HORZ, ,
   !REXXDW.!DW_EXPAND_VERT, 0
...
win = dw_box_unpack_at_index( box, index )
```

6.1.2. Widgets/Button [Modules]

[Top] [Widgets] [Modules]

DESCRIPTION

Buttons enable an action to be carried out by the user clicking the left mouse button on the widget. RexxDW supports two types of button; plain text buttons and bitmap buttons.

6.1.2.1. Button/DW_button_new [Functions]

[Top] [Button] [Functions]

NAME

DW_button_new

SYNOPSIS

win = dw_button_new(ButtonText, Id)

FUNCTION

Creates a new text button. A button is a widget which contains text and can be clicked to execute some action.

ARGUMENTS

- ButtonText - the text to display in the button
- Id - a numeric identifier used to identify the window

RESULT

A window identifier.

SEE ALSO

DW_bitmapbutton_new_from_file, DW_bitmapbutton_new_from_data

SOURCE

```
...
button = dw_button_new( "Quit", 10 )
```

6.1.2.2. Button/DW_bitmapbutton_new_from_file [Functions]

[Top] [Button] [Functions]

NAME

DW_bitmapbutton_new_from_file

SYNOPSIS

win = dw_bitmapbutton_new_from_file(BubbleText, Id, Filename)

FUNCTION

Creates a new button window using a bitmap image.

ARGUMENTS

- BubbleText - the text to display when the mouse is over the button
- Id - a numeric identifier used to identify the window
- Filename - the name of a file containing a valid bitmap image (.BMP, or .XPM)

RESULT

A window identifier.

SEE ALSO

DW_button_new, DW_bitmapbutton_new_from_data

SOURCE

```
...
button = dw_bitmapbutton_new_from_file( 'Quit', 0, 'quit' )
Call dw_box_pack_start win, button, 32, 32, !REXXDW.!DW_DONT_EXPAND_HORZ, ,
   !REXXDW.!DW_DONT_EXPAND_VERT, 0
```

6.1.2.3. Button/DW_bitmapbutton_new_from_data [Functions]

[Top] [Button] [Functions]

NAME

DW_bitmapbutton_new_from_data

SYNOPSIS

win = dw_bitmapbutton_new_from_data(BubbleText, Id, Data)

FUNCTION

Creates a new button window using a bitmap image.

ARGUMENTS

- BubbleText - the text to display when the mouse is over the button
- Id - a numeric identifier used to identify the window
- Data - the exact contents of the image.

RESULT

A window identifier.

SEE ALSO

DW_button_new, DW_bitmapbutton_new_from_file

SOURCE

```
...
ico = "0000A6D7007E"
...
button = dw_bitmapbutton_new_from_data( 'Quit', 0, x2c(ico) )
Call dw_box_pack_start win, button, 32, 32, !REXXDW.!DW_DONT_EXPAND_HORZ, ,
   !REXXDW.!DW_DONT_EXPAND_VERT, 0
```

6.1.3. Widgets/Calendar [Modules]

[Top] [Widgets] [Modules]

DESCRIPTION

The **Calendar** widget displays a month calendar from which the user can select a date.

6.1.3.1. Calendar/DW_calendar_new [Functions]

[Top] [Calendar] [Functions]

NAME

DW_calendar_new

SYNOPSIS

win = dw_calendar_new(Id)

FUNCTION

Creates a new calendar. A calendar is a widget which allows the user to select a date.

ARGUMENTS

- Id - a numeric identifier used to identify the window

RESULT

A window identifier.

SEE ALSO

DW_calendar_get_date, DW_calendar_set_date

NOTES

On OS/2 this is a dummy text widget. On GTK, only 1 month is ever displayed. On Windows, more than 1 month is displayed depending on whether the widget is packed with expanding on.

SOURCE

```
...
calendar = dw_calendar_new( 10 )
```

6.1.3.2. Calendar/DW_calendar_get_date [Functions]

[Top] [Calendar] [Functions]

NAME

DW_calendar_get_date

SYNOPSIS

date = dw_calendar_get_date(Win)

FUNCTION

Gets the selected date of the widget.

ARGUMENTS

- Win - the window identifier returned from DW_calendar_new()

RESULT

The date selected in Rexx's DATE('S') format.

SEE ALSO

DW_calendar_new, DW_calendar_set_date

SOURCE

```
...
calendar = dw_calendar_new( 10 )
...
date = dw_calendar_get_date( calendar )
```

6.1.3.3. Calendar/DW_calendar_set_date [Functions]

[Top] [Calendar] [Functions]

NAME

DW_calendar_set_date

SYNOPSIS

dw_calendar_set_date(Win, Date)

FUNCTION

Sets the date for the widget.

ARGUMENTS

- Win - the window identifier returned from DW_calendar_new()
- Date - the date to set the widget to. Must be in Rexx DATE('S') format.

RESULT

None

SEE ALSO

DW_calendar_new, DW_calendar_get_date

SOURCE

```
...
calendar = dw_calendar_new( 10 )
...
date = dw_calendar_set_date( calendar, Date( 'S' ) )
```

6.1.4. Widgets/CheckboxAndRadiobutton [Modules]

[Top] [Widgets] [Modules]

DESCRIPTION

Checkboxes provide a simple mechanism for a user to indicate a boolean value response. Radiobuttons provide the user with a means to choose from a list of options.

6.1.4.1. CheckboxAndRadiobutton/DW_checkbox_new [Functions]

[Top] [CheckboxAndRadiobutton] [Functions]

NAME

DW_checkbox_new

SYNOPSIS

win = dw_checkbox_new(CheckboxText, Id)

FUNCTION

Creates a new checkbox. A checkbox is a widget that allows for a boolean value to be set. It contains text to describe the boolean value being set.

ARGUMENTS

- CheckboxText - the text to display next to the checkbox
- Id - a numeric identifier used to identify the window

RESULT

A window identifier.

SEE ALSO

DW_checkbox_get, DW_checkbox_set

SOURCE

```
...
checkbox = dw_checkbox_new( "Allow user to delete", 0 )
```

6.1.4.2. CheckboxAndRadiobutton/DW_radiobutton_new [Functions]

[Top] [CheckboxAndRadiobutton] [Functions]

NAME

DW_radiobutton_new

SYNOPSIS

win = dw_radiobutton_new(ButtonText, Id)

FUNCTION

Creates a new radiobutton. A radiobutton is a widget that allows for one of a number of values to be set. It contains text to describe the value being set. A number of radiobuttons are created together with one of them set at any point in time.

ARGUMENTS

- ButtonText - the text to display next to the radiobutton
- Id - a numeric identifier used to identify the window

RESULT

A window identifier.

SEE ALSO

DW_checkbox_new, DW_groupbox_new

NOTES

For radiobuttons to work properly, all radiobuttons that are to be grouped together MUST be packed into the same groupbox and they MUST have a unique Id within the groupbox.

SOURCE

```
langs = 'Rexx Perl Tcl C COBOL'
groupbox = dw_groupbox_new( !REXXDW.!DW_VERT, 0, 'Favourite Language' )
Do i = 1 To Words( langs )
   rb.i = dw_radiobutton_new( Word( langs, i ), i*10 )
   Call dw_box_pack_start groupbox, rb.i, 150, 15, !REXXDW.!DW_EXPAND_HORZ, ,
      !REXXDW.!DW_DONT_EXPAND_VERT, 0
End
Call dw_radiobutton_set rb.1, !REXXDW.!DW_CHECKED
```

6.1.4.3. CheckboxAndRadiobutton/DW_checkbox_set [Functions]

[Top] [CheckboxAndRadiobutton] [Functions]

NAME

DW_checkbox_set

SYNOPSIS

dw_checkbox_set(Win, State)

FUNCTION

Sets the state of the specified checkbox.

ARGUMENTS

- Win - the window identifier returned from DW_checkbox_new()
- State - the initial state of the checkbox; !REXXDW.!DW_CHECKED or !REXXDW.!DW_UNCHECKED

RESULT

No return result.

SEE ALSO

DW_checkbox_new, DW_checkbox_get, WidgetChecked

SOURCE

```
...
checkbox = dw_checkbox_new( "Allow access", 0 )
Call dw_checkbox_set checkbox, !REXXDW.!DW_CHECKED
```

6.1.4.4. CheckboxAndRadiobutton/DW_radiobutton_set [Functions]

[Top] [CheckboxAndRadiobutton] [Functions]

NAME

DW_radiobutton_set

SYNOPSIS

dw_radiobutton_set(Win, State)

FUNCTION

Sets the state of the radiobutton to checked or not.

ARGUMENTS

- Win - the window identifier returned from DW_radiobutton_new()
- State - set to DW_CHECKED or DW_UNCHECKED

RESULT

No return result.

SEE ALSO

DW_radiobutton_get

SOURCE

```
langs = 'Rexx Perl Tcl C COBOL'
groupbox = dw_groupbox_new( !REXXDW.!DW_VERT, 0, 'Favourite Language' )
Do i = 1 To Words( langs )
   rb.i = dw_radiobutton_new( Word( langs, i ), i*10 )
   Call dw_box_pack_start groupbox, rb.i, 150, 15, !REXXDW.!DW_EXPAND_HORZ, ,
      !REXXDW.!DW_DONT_EXPAND_VERT, 0
End
Call dw_radiobutton_set rb.1, !REXXDW.!DW_CHECKED
```

6.1.4.5. CheckboxAndRadiobutton/DW_checkbox_get [Functions]

[Top] [CheckboxAndRadiobutton] [Functions]

NAME

DW_checkbox_get

SYNOPSIS

win = dw_checkbox_get(Win)

FUNCTION

Queries the state of the specified checkbox.

ARGUMENTS

- Win - the window identifier returned from DW_checkbox_new()

RESULT

1 if the checkbox is checked, 0 if not checked.

SEE ALSO

DW_checkbox_new, DW_checkbox_set

SOURCE

```
...
checkbox = dw_checkbox_new( "Allow access", 0 )
...
rc = dw_checkbox_get( checkbox )
cb_result = 'Unchecked Checked'
Say 'Allow access checkbox' Word( cb_result, rc+1 )
```

6.1.4.6. CheckboxAndRadiobutton/DW_radiobutton_get [Functions]

[Top] [CheckboxAndRadiobutton] [Functions]

NAME

DW_radiobutton_get

SYNOPSIS

boolean = dw_radiobutton_get(Win)

FUNCTION

Determines if a radiobutton has been checked or not.

ARGUMENTS

- Win - the window identifier returned from DW_radiobutton_new()

RESULT

1 if the radiobutton is checked, 0 if unchecked

SEE ALSO

DW_radiobutton_set

SOURCE

```
langs = 'Rexx Perl Tcl C COBOL'
groupbox = dw_groupbox_new( !REXXDW.!DW_VERT, 0, 'Favourite Language' )
Do i = 1 To Words( langs )
   rb.i = dw_radiobutton_new( Word( langs, i ), i*10 )
   Call dw_box_pack_start groupbox, rb.i, 150, 15, !REXXDW.!DW_EXPAND_HORZ, ,
      !REXXDW.!DW_DONT_EXPAND_VERT, 0
End
Call dw_radiobutton_set rb.1, !REXXDW.!DW_CHECKED
...
Do i = 1 To Words( langs )
   If dw_radiobutton_get( rb.i ) Then
      Do
         Say 'Your favourite language is' Word( langs, i )
         Leave
      End
End
```

6.1.5. Widgets/ContainerAndFilesystem [Modules]

[Top] [Widgets] [Modules]

DESCRIPTION

These widgets provide a two-dimensional display of read-only data. Data are displayed in rows and columns, with automatically created scrollbars to display the complete set of data. Columns can be resized by the user at runtime. A Filesystem is the same as a normal Container, except that the first column consists of a column of string and icon items, which usually displays a filename.

All DW_container*() functions (for which there is no specific "Filename" equivalent) also works with a "Filename" widget.

6.1.5.1. ContainerAndFilesystem/DW_container_new [Functions]

[Top] [ContainerAndFilesystem] [Functions]

NAME

DW_container_new

SYNOPSIS

win = dw_container_new(Id, Selection)

FUNCTION

Creates a new container. A container is a widget that provides a grid of cells with column headings.

ARGUMENTS

- Id - a numeric identifier used to identify the window
- Selection - !REXXDW.!DW_SINGLE_SELECTION indicates that only 1 row in the container is selectable !REXXDW.!DW_MULTIPLE_SELECTION indicates that multiple rowsin the container are selectable

RESULT

A window identifier.

SEE ALSO

ContainerSelection

SOURCE

```
...
container = dw_container_new( 1, !REXXDW.!DW_SINGLE_SELECTION )
```

6.1.5.2. ContainerAndFilesystem/DW_container_alloc [Functions]

[Top] [ContainerAndFilesystem] [Functions]

NAME

DW_container_alloc

SYNOPSIS

alloc = dw_container_alloc(Win, RowCount)

FUNCTION

Allocates working memory for a container window.

ARGUMENTS

- Win - the window identifier returned from DW_container_new()
- RowCount - the number of rows of data that the container will initially contain

RESULT

An identifier for the working memory.

SEE ALSO

DW_container_new, DW_container_clear

NOTES

After a container window is created, you need to allocate memory for the initial number of rows that the container will contain. This call does that and must be called after creating the container. The allocated memory will be freed when the container is destroyed and by a call to DW_container_clear().

SOURCE

```
...
container = dw_container_new( 1, !REXXDW.!DW_SINGLE_SELECTION )
alloc = dw_container_alloc( container, 10 )
...
```

6.1.5.3. ContainerAndFilesystem/DW_container_setup [Functions]

[Top] [ContainerAndFilesystem] [Functions]

NAME

DW_container_setup

SYNOPSIS

dw_container_setup(Win, FlagsArray, TitlesArray[,Separator])

FUNCTION

Defines properties for the columns in a container.

ARGUMENTS

- Win - the window identifier returned from DW_container_new()
- FlagsArray - an array containing column flags
- TitlesArray- an array containing column heading titles
- Separator - An optional flag under OS/2 ?????

RESULT

No return result.

SEE ALSO

DW_container_change_item, DW_filesystem_setup, ContainerColumnFlags

NOTES

The FlagsArray is a Rexx stem variable name, including a trailing period. The 0th compound variable (eg Flags.0) contains a count of the number of items in the array. Each item in the array can contain a combination of the values described in ContainerColumnFlags.

SOURCE

```
...
flags.0 = 2
flags.1 = dw_or( !REXXDW.!DW_CFA_STRINGANDICON, !REXXDW.!DW_CFA_LEFT, ,
   !REXXDW.!DW_CFA_SEPARATOR, !REXXDW.!DW_CFA_HORZSEPARATOR )
flags.2 = dw_or( !REXXDW.!DW_CFA_STRING, !REXXDW.!DW_CFA_LEFT, ,
   !REXXDW.!DW_CFA_SEPARATOR, !REXXDW.!DW_CFA_HORZSEPARATOR )
titles.0 = 2
titles.1 = 'Title'
titles.2 = 'Command'
...
container = dw_container_new( 1, !REXXDW.!DW_MULTIPLE_SELECTION )
Call dw_container_setup container, 'flags.', 'titles.', 0
alloc = dw_container_alloc( container, numrows )
...
Do i = 1 To numrows
   Call dw_container_set_item container, alloc, 0, i-1, title.i, icon.i
   Call dw_container_set_item container, alloc, 1, i-1, command.i
   Call dw_container_set_row_title alloc, i-1, "Row"i
End
```

6.1.5.4. ContainerAndFilesystem/DW_filesystem_setup [Functions]

[Top] [ContainerAndFilesystem] [Functions]

NAME

DW_filesystem_setup

SYNOPSIS

dw_filesystem_setup(Win, FlagsArray, TitlesArray, Separator)

FUNCTION

Defines properties for the columns in a filesystem container.

ARGUMENTS

- Win - the window identifier returned from DW_container_new()
- FlagsArray - an array containing column flags
- TitlesArray- an array containing column heading titles

RESULT

No return result.

SEE ALSO

DW_filesystem_change_item, DW_container_setup, ContainerColumnFlags

NOTES

The FlagsArray is a Rexx stem variable name, including a trailing period. The 0th compound variable (eg Flags.0) contains a count of the number of items in the array. Each item in the array can contain a combination of the values described in ContainerColumnFlags.

SOURCE

```
...
flags.0 = 2
flags.1 = dw_or( !REXXDW.!DW_CFA_STRINGANDICON, !REXXDW.!DW_CFA_LEFT, ,
   !REXXDW.!DW_CFA_SEPARATOR, !REXXDW.!DW_CFA_HORZSEPARATOR )
flags.2 = dw_or( !REXXDW.!DW_CFA_STRING, !REXXDW.!DW_CFA_LEFT, ,
   !REXXDW.!DW_CFA_SEPARATOR, !REXXDW.!DW_CFA_HORZSEPARATOR )
titles.0 = 2
titles.1 = 'Title'
titles.2 = 'Command'
...
container = dw_container_new( 1, !REXXDW.!DW_MULTIPLE_SELECTION )
alloc = dw_container_alloc( container, numrows )
Call dw_filesystem_setup container, 'flags.', 'titles.'
...
normalfileicon = dw_icon_load_from_file( '/home/mark/normalfile' )
Do i = 1 To numrows
   Call dw_filesystem_set_file container, alloc, '/home/mark/myfile', myicon
   Call dw_filesystem_set_item container, alloc, 0, i-1, title.i, icon.i
   Call dw_filesystem_set_item container, alloc, 1, i-1, command.i
   Call dw_container_set_row_title alloc, i-1, "Row"i
End
```

6.1.5.5. ContainerAndFilesystem/DW_container_set_item [Functions]

[Top] [ContainerAndFilesystem] [Functions]

NAME

DW_container_set_item

SYNOPSIS

dw_container_set_item(Win, Memory, Column, Row, Data [,Data2])

FUNCTION

Adds the initial data into a cell of a container.

ARGUMENTS

- Win - the window identifier returned from DW_container_new()
- Memory - the memory identifier returned by a call to DW_container_alloc()
- Column - the column of the data to be changed, 0 based
- Row - the row of the data to be changed, 0 based
- Data - the new data value for the cell

- Data2 - extra data for the cell if the column type is !REXXDW.!DW_CFA_STRINGANDICON (not implemented yet)

RESULT

No return result.

SEE ALSO

DW_container_change_item, DW_container_setup, ContainerColumnFlags DW_filesystem_change_item, DW_filesystem_set_item

NOTES

The Data (and Data2) arguments must match the data type of the column when defined in the call to DW_container_setup(). To clear a displayed icon in a column of type DW_CFA_BITMAPORICON set Data argument to 0.

SOURCE

```
...
flags.0 = 2
flags.1 = dw_or( !REXXDW.!DW_CFA_STRINGANDICON, !REXXDW.!DW_CFA_LEFT, ,
   !REXXDW.!DW_CFA_SEPARATOR, !REXXDW.!DW_CFA_HORZSEPARATOR )
flags.2 = dw_or( !REXXDW.!DW_CFA_STRING, !REXXDW.!DW_CFA_LEFT, ,
   !REXXDW.!DW_CFA_SEPARATOR, !REXXDW.!DW_CFA_HORZSEPARATOR )
titles.0 = 2
titles.1 = 'Title'
titles.2 = 'Command'
...
container = dw_container_new( 1, !REXXDW.!DW_MULTIPLE_SELECTION )
alloc = dw_container_alloc( container, numrows )
Call dw_container_setup container, 'flags.', 'titles.', 0
...
Do i = 1 To numrows
   Call dw_container_set_item container, alloc, 0, i-1, title.i, icon.i
   Call dw_container_set_item container, alloc, 1, i-1, command.i
   Call dw_container_set_row_title alloc, i-1, "Row"i
End
```

6.1.5.6. ContainerAndFilesystem/DW_filesystem_set_file [Functions]

[Top] [ContainerAndFilesystem] [Functions]

NAME

DW_filesystem_set_file

SYNOPSIS

dw_filesystem_set_file(Win, Memory, Row, Filename, Icon)

FUNCTION

Adds the data in the first cell of a filesystem container.

ARGUMENTS

- Win - the window identifier returned from DW_container_new()
- Memory - the memory identifier returned by a call to DW_container_alloc()
- Row - the row of the data to be changed, 0 based
- Filename - the new filename for the cell
- Icon - the icon identifier returned from DW_icon_load_from_file() or DW_icon_load_from_data() to be the new icon for the cell

RESULT

No return result.

SEE ALSO

DW_filesystem_change_file, DW_filesystem_set_item

SOURCE

```
...
normalfileicon = dw_icon_load_from_file( '/home/mark/normalfile' )
container = dw_container_new( 1, !REXXDW.!DW_SINGLE_SELECTION )
alloc = dw_container_alloc( container, 10 )
...
Call dw_filesystem_set_file container, alloc, 10, '/home/mark/newfile.txt', ,
   normalfileicon
```

6.1.5.7. ContainerAndFilesystem/DW_filesystem_set_item [Functions]

[Top] [ContainerAndFilesystem] [Functions]

NAME

DW_filesystem_set_item

SYNOPSIS

dw_filesystem_set_item(Win, Memory, Column, Row, Data [,Data2])

FUNCTION

Adds the initial data into a cell of a filesystem container.

ARGUMENTS

- Win - the window identifier returned from DW_container_new()

- Memory - the memory identifier returned by a call to DW_container_alloc()
- Column - the column of the data to be changed, 0 based
- Row - the row of the data to be changed, 0 based
- Data - the new data value for the cell
- Data2 - extra data for the cell if the column type is !REXXDW.!DW_CFA_STRINGANDICON (not implemented yet)

RESULT

No return result.

SEE ALSO

DW_container_change_item, DW_container_setup, ContainerColumnFlags DW_filesystem_change_item, DW_container_set_item

NOTES

The Data (and Data2) arguments must match the data type of the column when defined in the call to DW_container_setup().

SOURCE

```
...
flags.0 = 2
flags.1 = dw_or( !REXXDW.!DW_CFA_STRINGANDICON, !REXXDW.!DW_CFA_LEFT, ,
   !REXXDW.!DW_CFA_SEPARATOR, !REXXDW.!DW_CFA_HORZSEPARATOR )
flags.2 = dw_or( !REXXDW.!DW_CFA_STRING, !REXXDW.!DW_CFA_LEFT, ,
   !REXXDW.!DW_CFA_SEPARATOR, !REXXDW.!DW_CFA_HORZSEPARATOR )
titles.0 = 2
titles.1 = 'Title'
titles.2 = 'Command'
...
container = dw_container_new( 1, !REXXDW.!DW_MULTIPLE_SELECTION )
alloc = dw_container_alloc( container, numrows )
Call dw_filesystem_setup container, 'flags.', 'titles.', 0
...
Do i = 1 To numrows
   Call dw_filesystem_set_file container, alloc, i-1, fname.i, icon.i
   Call dw_filesystem_set_item container, alloc, 0, i-1, title.i, icon.i
   Call dw_filesystem_set_item container, alloc, 1, i-1, command.i
   Call dw_container_set_row_title alloc, i-1, "Row"i
End
```

6.1.5.8. ContainerAndFilesystem/DW_container_change_item [Functions]

[Top] [ContainerAndFilesystem] [Functions]

NAME

DW_container_change_item

SYNOPSIS

dw_container_change_item(Win, Column, Row, Data [,Data2])

FUNCTION

Changes the data in a cell of a container.

ARGUMENTS

- Win - the window identifier returned from DW_container_new()
- Column - the column of the data to be changed, 0 based
- Row - the row of the data to be changed, 0 based
- Data - the new data value for the cell
- Data2 - extra data for the cell if the column type is !REXXDW.!DW_CFA_STRINGANDICON (not implemented yet)

RESULT

No return result.

SEE ALSO

DW_container_set_item

NOTES

The Data (and Data2) arguments must match the data type of the column when defined in the call to DW_container_setup(). To clear a displayed icon in a column of type DW_CFA_BITMAPORICON set Data argument to 0.

SOURCE

```
...
container = dw_container_new( 1, !REXXDW.!DW_SINGLE_SELECTION )
...
Call dw_container_change_item container, 0, 10, 'New string data'
```

6.1.5.9. ContainerAndFilesystem/DW_filesystem_change_file [Functions]

[Top] [ContainerAndFilesystem] [Functions]

NAME

DW_filesystem_change_file

SYNOPSIS

dw_filesystem_change_file(Win, Row, Filename, Icon)

FUNCTION

Changes the data in the first cell of a filesystem container.

ARGUMENTS

- Win - the window identifier returned from DW_container_new()
- Row - the row of the data to be changed, 0 based
- Filename - the new filename for the cell
- Icon - the icon identifier returned from DW_icon_load_from_file() or DW_icon_load_from_data() to be the new icon for the cell

RESULT

No return result.

SEE ALSO

DW_filesystem_change_item, DW_filesystem_set_file

SOURCE

```
...
normalfileicon = dw_icon_load_from_file( '/home/mark/normalfile' )
...
container = dw_container_new( 1, !REXXDW.!DW_SINGLE_SELECTION )
...
Call dw_filesystem_change_file container, 10, '/home/mark/newfile.txt', ,
   normalfileicon
```

6.1.5.10. ContainerAndFilesystem/DW_filesystem_change_item [Functions]

[Top] [ContainerAndFilesystem] [Functions]

NAME

DW_filesystem_change_item

SYNOPSIS

dw_filesystem_change_item(Win, Column, Row, Data [,Data2])

FUNCTION

Changes the data in a cell of a container.

ARGUMENTS

- Win - the window identifier returned from DW_container_new()
- Column - the column of the data to be changed, 0 based

- Row - the row of the data to be changed, 0 based
- Data - the new data value for the cell
- Data2 - extra data for the cell if the column type is !REXXDW.!DW_CFA_STRINGANDICON (not implemented yet)

RESULT

No return result.

SEE ALSO

DW_container_set_item

NOTES

The Data (and Data2) arguments must match the data type of the column when defined in the call to DW_container_setup().

SOURCE

```
...
container = dw_container_new( 1, !REXXDW.!DW_SINGLE_SELECTION )
...
Call dw_filesystem_change_item container, 0, 10, 'New string data'
```

6.1.5.11. ContainerAndFilesystem/DW_container_insert [Functions]

[Top] [ContainerAndFilesystem] [Functions]

NAME

DW_container_insert

SYNOPSIS

dw_container_insert(Win, Memory, RowCount)

FUNCTION

Inserts the allocated memory into the container.

ARGUMENTS

- Win - the window identifier returned from DW_container_new()
- Memory - the memory identifier returned by a call to DW_container_alloc()
- RowCount - the number of rows of data that the container will initially contain

RESULT

No return result.

SEE ALSO

DW_container_new, DW_container_alloc

SOURCE

```
...
container = dw_container_new( 1, !REXXDW.!DW_SINGLE_SELECTION )
alloc = dw_container_alloc( container, numrows )
...
Call dw_container_insert container, alloc, numrows
```

6.1.5.12. ContainerAndFilesystem/DW_container_set_row_title [Functions]

[Top] [ContainerAndFilesystem] [Functions]

NAME

DW_container_set_row_title

SYNOPSIS

dw_container_set_row_title(Memory, Row, TitleText)

FUNCTION

Assigns a textual string for the specified Row. Used to identify this row in other container functions

ARGUMENTS

- Memory - the memory identifier returned by a call to DW_container_alloc()
- Row - the row of data to be assigned the value; 0 based
- TitleText - the text to be associated with the row

RESULT

No return result.

SEE ALSO

DW_container_cursor, DW_container_delete_row, DW_container_query_next, DW_container_query_start

SOURCE

```
...
container = dw_container_new( 1, !REXXDW.!DW_SINGLE_SELECTION )
alloc = dw_container_alloc( container, numrows )
...
Do i = 1 To numrows
```

```
   Call dw_container_set_row_title alloc, i-1, "Row"i
End
...
Call dw_container_delete_row container, "Row10"
```

6.1.5.13. ContainerAndFilesystem/DW_container_set_stripe [Functions]

[Top] [ContainerAndFilesystem] [Functions]

NAME

DW_container_set_stripe

SYNOPSIS

dw_container_set_stripe(Win, OddColour, EvenColour)

FUNCTION

Sets the background colours for alternate rows in a container.

ARGUMENTS

- Win - the window identifier returned from DW_container_new()
- OddColour - the internal RexxDW colour for odd numbered rows
- EvenColour - the internal RexxDW colour for even numbered rows

RESULT

No return result.

SEE ALSO

Colours, DW_rgb

SOURCE

```
...
container = dw_container_new( 1, !REXXDW.!DW_SINGLE_SELECTION )
alloc = dw_container_alloc( container, numrows )
Call dw_container_set_stripe container, !REXXDW.!DW_CLR_YELLOW, !REXXDW.!DW_CLR_DEFAULT
...
```

6.1.5.14. ContainerAndFilesystem/DW_container_set_column_width [Functions]

[Top] [ContainerAndFilesystem] [Functions]

NAME

DW_container_set_column_width

SYNOPSIS

dw_container_set_column_width(Win, Column, Width)

FUNCTION

Sets the width of the column in the container to Width pixels.

ARGUMENTS

- Win - the window identifier returned from DW_container_new()
- Column - the column that requires a specific width; 0 based
- Width - the width of the column in pixels

RESULT

No return result.

SEE ALSO

DW_container_optimize

NOTES

Columns with a set fixed width are not changed when DW_container_optimize() is called.

SOURCE

```
...
container = dw_container_new( 1, !REXXDW.!DW_MULTIPLE_SELECTION )
...
Call dw_container_set_width( container, 0, 25 )
```

6.1.5.15. ContainerAndFilesystem/DW_container_optimize [Functions]

[Top] [ContainerAndFilesystem] [Functions]

NAME

DW_container_optimize

SYNOPSIS

dw_container_optimize(Win)

FUNCTION

Rearranges the width of the container columns to provide the optimum display of the columns.

ARGUMENTS

- Win - the window identifier returned from DW_container_new()

RESULT

No return result.

SOURCE

```
...
container = dw_container_new( 1, !REXXDW.!DW_SINGLE_SELECTION )
alloc = dw_container_alloc( container, numrows )
...
Call dw_container_insert container, alloc, numrows
Call dw_container_optimize container
```

6.1.5.16. ContainerAndFilesystem/DW_container_clear [Functions]

[Top] [ContainerAndFilesystem] [Functions]

NAME

DW_container_clear

SYNOPSIS

dw_container_clear(Win, Redraw)

FUNCTION

Clears the container of all rows and frees the memory allocated with DW_container_alloc().

ARGUMENTS

- Win - the window identifier returned from DW_container_new()
- Redraw - see ContainerClearFlags

RESULT

No return result.

SEE ALSO

DW_container_delete, ContainerClearFlags

SOURCE

```
...
container = dw_container_new( 1, !REXXDW.!DW_SINGLE_SELECTION )
...
Call dw_container_clear container, !REXXDW.!DW_DONT_REDRAW
```

6.1.5.17. ContainerAndFilesystem/DW_container_delete [Functions]

[Top] [ContainerAndFilesystem] [Functions]

NAME

DW_container_delete

SYNOPSIS

dw_container_delete(Win, NumRows)

FUNCTION

Deletes NumRows from the the container starting with the first row.

ARGUMENTS

- Win - the window identifier returned from DW_container_new()
- NumRows - the number of rows to delete

RESULT

No return result.

SEE ALSO

DW_container_delete_row

SOURCE

```
...
container = dw_container_new( 1, !REXXDW.!DW_SINGLE_SELECTION )
...
Call dw_container_delete container, 5
```

6.1.5.18. ContainerAndFilesystem/DW_container_delete_row [Functions]

[Top] [ContainerAndFilesystem] [Functions]

NAME

DW_container_delete_row

SYNOPSIS

dw_container_delete_row(Win, TitleText)

FUNCTION

Deletes the row from the container which has the specified TitleText

ARGUMENTS

- Win - the window identifier returned from DW_container_new()
- TitleText - the text that was set by a call to DW_container_set_row_title()

RESULT

No return result.

SEE ALSO

DW_container_delete, DW_container_set_row_title

SOURCE

```
...
container = dw_container_new( 1, !REXXDW.!DW_SINGLE_SELECTION )
alloc = dw_container_alloc( container, numrows )
...
Do i = 1 To numrows
   Call dw_container_set_row_title alloc, i-1, "Row"i
End
...
Call dw_container_delete_row container, "Row10"
```

6.1.5.19. ContainerAndFilesystem/DW_container_query_start [Functions]

[Top] [ContainerAndFilesystem] [Functions]

NAME

DW_container_query_start

SYNOPSIS

text = dw_container_query_start(Win, Flags)

FUNCTION

Initiates a query on a container for subsequent calls to DW_container_query_next(). Returns the title text associated with the first row in the container based on Flags. The title text having previously been set with a call to DW_container_set_row_title().

ARGUMENTS

- Win - the window identifier returned from DW_container_new()
- Flags - !REXXDW.!DW_CRA_ALL indicates that all rows in the container should be queried. !REXXDW.!DW_CRA_SELECTED indicates that only those rows that have been selected should be queried. !REXXDW.!DW_CRA_CURSORED indicates that only those items that have focus should be queried.

RESULT

The title text of the row or the empty string to indicate the end of the query. ie no more rows are available for querying.

SEE ALSO

DW_container_query_next, DW_container_set_row_title

SOURCE

```
...
container = dw_container_new( 1, !REXXDW.!DW_MULTIPLE_SELECTION )
...
selected = dw_container_query_start( container, !REXXDW.!DW_CRA_SELECTED )
Do While selected \= ''
   Say 'Title of row selected is' selected
   selected = dw_container_query_next( container, !REXXDW.!DW_CRA_SELECTED )
End
```

6.1.5.20. ContainerAndFilesystem/DW_container_query_next [Functions]

[Top] [ContainerAndFilesystem] [Functions]

NAME

DW_container_query_next

SYNOPSIS

text = dw_container_query_next(Win, Flags)

FUNCTION

Returns the title text associated with the next row in the container. The title text having previously been set with a call to DW_container_set_row_title().

ARGUMENTS

- Win - the window identifier returned from DW_container_new()
- Flags - !REXXDW.!DW_CRA_ALL indicates that all rows in the container should be queried. !REXXDW.!DW_CRA_SELECTED indicates that only those rows that have been selected should be

queried. !REXXDW.!DW_CRA_CURSORED indicates that only those items that have focus should be queried.

RESULT

The title text of the row or the empty string to indicate the end of the query. ie no more rows are available for querying.

NOTES

As the only way of knowing that the query has returned all rows in subsequent calls to this function, all rows MUST have a non-blank title text.

SEE ALSO

DW_container_query_start, DW_container_set_row_title

SOURCE

```
...
container = dw_container_new( 1, !REXXDW.!DW_MULTIPLE_SELECTION )
...
selected = dw_container_query_start( container, !REXXDW.!DW_CRA_SELECTED )
Do While selected \= ''
   Say 'Title of row selected is' selected
   selected = dw_container_query_next( container, !REXXDW.!DW_CRA_SELECTED )
End
```

6.1.5.21. ContainerAndFilesystem/DW_container_cursor [Functions]

[Top] [ContainerAndFilesystem] [Functions]

NAME

DW_container_cursor

SYNOPSIS

dw_container_cursor(Win, TitleText)

FUNCTION

Rearranges the view of the container so that the row with the specified TitleText is visible.

ARGUMENTS

- Win - the window identifier returned from DW_container_new()
- TitleText - the text that was set by a call to DW_container_set_row_title()

RESULT

No return result.

SEE ALSO

DW_container_set_row_title

SOURCE

```
...
container = dw_container_new( 1, !REXXDW.!DW_SINGLE_SELECTION )
alloc = dw_container_alloc( container, numrows )
...
Do i = 1 To numrows
   Call dw_container_set_row_title alloc, i-1, "Row"i
End
...
Call dw_container_cursor container, "Row10"
```

6.1.5.22. ContainerAndFilesystem/DW_container_scroll [Functions]

[Top] [ContainerAndFilesystem] [Functions]

NAME

DW_container_scroll

SYNOPSIS

dw_container_scroll(Win, Direction, NumRows)

FUNCTION

Scrolls the current view of the container by NumRows rows in the direction specified by Direction.

ARGUMENTS

- Win - the window identifier returned from DW_container_new()
- Direction: o !REXXDW.!DW_SCROLL_TOP shows the first rows in the container o !REXXDW.!DW_SCROLL_BOTTOM shows the last rows in the container o !REXXDW.!DW_SCROLL_UP scrolls up NumRows rows o !REXXDW.!DW_SCROLL_DOWN scrolls down NumRows rows

RESULT

No return result.

SEE ALSO

ContainerScrollFlags

NOTES

The NumRows argument is ignored for !REXXDW.!DW_SCROLL_TOP and !REXXDW.!DW_SCROLL_BOTTOM.

SOURCE

```
...
container = dw_container_new( 1, !REXXDW.!DW_MULTIPLE_SELECTION )
...
Call dw_container_scroll( container, !REXXDW.!DW_SCROLL_UP, 10 )
```

6.1.6. Widgets/Rendering [Modules]

[Top] [Widgets] [Modules]

DESCRIPTION

A rendering widget allows low-level graphics drawing to be done. Two widget types can be rendered; a render window and a pixmap.

The "normal" way to write low-level graphics is to draw into an off-screen pixmap, and then "bitblt" the off-screen pixmap into the render window. This is much faster that drawing directly into the render window itself.

NOTES

Two callbacks can be used with Render windows; DW_EXPOSE_EVENT and DW_CONFIGURE_EVENT. Whenever the Render window's size is changed a "configure" event is fired. Whenever a portion of the window is made visible, an "expose" event fires.

6.1.6.1. Rendering/DW_render_new [Functions]

[Top] [Rendering] [Functions]

NAME

DW_render_new

SYNOPSIS

win = dw_render_new(Id)

FUNCTION

Creates a new render window. A render window is used to draw primitive graphics on, or to be the window in which a pixmap is rendered.

ARGUMENTS

- Id - a numeric identifier used to identify the window

RESULT

A render identifier.

SOURCE

```
render = dw_render_new( 10 )
depth = dw_color_depth_get()
pixmap = dw_pixmap_new( render, 64, 64, depth )
```

6.1.6.2. Rendering/DW_pixmap_new [Functions]

[Top] [Rendering] [Functions]

NAME

DW_pixmap_new

SYNOPSIS

win = dw_pixmap_new(Win, Width, Height, Depth)

FUNCTION

Creates a new pixmap widget of the given dimensions.

ARGUMENTS

- Win - the window identifier returned from DW_render_new() that the pixmap is associated with (or zero)
- Width - the width of the pixmap in pixels
- Height - the height of the pixmap in pixels
- Depth - the colour depth of the pixmap as returned from DW_color_depth_get()

RESULT

A pixmap identifier.

NOTES

The Win argument can be specified as 0. This will result in the pixmap being written "off screen". This is usually done to avoid flicker if several drawing actions are done to the pixmap before rendering it to the screen.

SOURCE

```
render = dw_render_new( 10 )
depth = dw_color_depth_get()
pixmap = dw_pixmap_new( render, 64, 64, depth )
```

6.1.6.3. Rendering/DW_pixmap_new_from_file [Functions]

[Top] [Rendering] [Functions]

NAME

DW_pixmap_new_from_file

SYNOPSIS

win = dw_pixmap_new_from_file(Win, Filename)

FUNCTION

Creates a new pixmap widget from the specified Filename.

ARGUMENTS

- Win - the window identifier returned from DW_render_new() that the pixmap is associated with (or zero)
- Filename - the name of a file containing a valid image

RESULT

A pixmap identifier.

SEE ALSO

DW_pixmap_new_from_data

NOTES

Under Windows and OS/2 a Bitmap file (.BMP) must be presented. Under Linux various image file formats are valid; eg. XPM, PNG, JPG The size of the pixmap is determined from the dimensions of the image. The Win argument can be specified as 0. This will result in the pixmap being written "off screen". This is usually done to avoid to avoid flicker if several drawing actions are done to the pixmap before rendering it to the screen.

SOURCE

```
render = dw_render_new( 10 )
pixmap = dw_pixmap_new_from_file( render, '/home/mark/mypixmap' )
```

6.1.6.4. Rendering/DW_pixmap_new_from_data [Functions]

[Top] [Rendering] [Functions]

NAME

DW_pixmap_new_from_data

SYNOPSIS

win = dw_pixmap_new_from_data(Win, Filename)

FUNCTION

Creates a new pixmap widget from the specified Filename.

ARGUMENTS

- Win - the window identifier returned from DW_render_new() that the pixmap is associated with (or zero)
- Data - the exact contents of the image.

RESULT

A pixmap identifier.

SEE ALSO

DW_pixmap_new_from_file

NOTES

Under Windows and OS/2 a Bitmap file (.BMP) must be presented. Under Linux various image file formats are valid; eg. XPM, PNG, JPG The size of the pixmap is determined from the dimensions of the image. The Win argument can be specified as 0. This will result in the pixmap being written "off screen". This is usually done to avoid to avoid flicker if several drawing actions are done to the pixmap before rendering it to the screen.

SOURCE

```
...
bmp = "0000A6D7007E"
...
render = dw_render_new( 10 )
pixmap = dw_pixmap_new_from_data( render, x2c( bmp ) )
```

6.1.6.5. Rendering/DW_pixmap_set_transparent_color [Functions]

[Top] [Rendering] [Functions]

NAME

DW_pixmap_set_transparent_color

SYNOPSIS

DW_pixmap_set_transparent_color(Pixmap, Color)

FUNCTION

Specifies the color to be used as the transparent background when this pixmap is bitblt'ed

ARGUMENTS

- Pixmap - the pixmap identifier returned from DW_pixmap_new(), DW_pixmap_new_from_file() or DW_pixmap_new_from_data.
- Color - - the color to be rendered transparent.

RESULT

No return result.

SEE ALSO

DW_pixmap_new, DW_pixmap_new_from_file, DW_pixmap_new_from_data

SOURCE

```
...
bmp = "0000A6D7007E"
...
render = dw_render_new( 10 )
pixmap = dw_pixmap_new_from_data( render, x2c( bmp ) )
Call dw_pixmap_set_transparent_color pixmap, !REXXDW.!DW_RED
```

6.1.6.6. Rendering/DW_pixmap_destroy [Functions]

[Top] [Rendering] [Functions]

NAME

DW_pixmap_destroy

SYNOPSIS

dw_pixmap_destroy(Pixmap)

FUNCTION

Destroys a Pixmap.

ARGUMENTS

- Pixmap - the pixmap identifier returned from DW_pixmap_new(), DW_pixmap_new_from_file() or DW_pixmap_new_from_data.

RESULT

No return result.

SEE ALSO

DW_pixmap_new, DW_pixmap_new_from_file, DW_pixmap_new_from_data

SOURCE

```
render = dw_render_new( 10 )
pixmap = dw_pixmap_new_from_file( render, '/home/mark/mypixmap' )
...
Call dw_pixmap_destroy( pixmap )
```

6.1.6.7. Rendering/DW_pixmap_bitblt [Functions]

[Top] [Rendering] [Functions]

NAME

DW_pixmap_bitblt

SYNOPSIS

dw_pixmap_bitblt(DestWin, DestPixmap, DestX, DestY, Width, Height, SrcWin, SrcPixmap, SrcX, SrcY)

FUNCTION

Copies a portion of a render window or pixmap to another render window or pixmap.

ARGUMENTS

- DestWin - the destination window identifier returned from DW_render_new()
- DestPixmap - the destination pixmap identifier returned from DW_pixmap_new(), DW_pixmap_new_from_file() or DW_pixmap_new_from_data
- DestX - the X coordinate of the destination measured in pixels
- DestY - the Y coordinate of the destination measured in pixels
- Width - the width of the portion to copy measured in pixels
- Height - the height of the portion to copy measured in pixels
- SrcWin - the source window identifier returned from DW_render_new()
- SrcPixmap - the source pixmap identifier returned from DW_pixmap_new()
- SrcX - the X coordinate of the source measured in pixels
- SrcY - the Y coordinate of the source measured in pixels

RESULT

No return result.

SEE ALSO

DW_pixmap_new, DW_pixmap_new_from_file, DW_pixmap_new_from_data, DW_render_new

NOTES

Only one of DestWin, DestPixmap is specified; the other should be 0 Only one of SrcWin, SrcPixmap is specified; the other should be 0

SOURCE

```
render = dw_render_new( 10 )
depth = dw_color_depth_get()
pixmap = dw_pixmap_new( render, 64, 64, depth )
...
Call dw_pixmap_bitblt( render, 0, 0, 0, 64, 64, 0, pixmap, 0, 0 )
```

6.1.6.8. Rendering/DW_pixmap_width [Functions]

[Top] [Rendering] [Functions]

NAME

DW_pixmap_width

SYNOPSIS

width = dw_pixmap_width(Win)

FUNCTION

returns the width of the specified pixmap.

ARGUMENTS

- Win - the window identifier returned from DW_pixmap_new(), DW_pixmap_new_from_file() or DW_pixmap_new_from_data().

RESULT

The pixmap's width

SEE ALSO

DW_pixmap_height

SOURCE

```
render = dw_render_new( 10 )
pixmap = dw_pixmap_new_from_file( render, '/home/mark/mypixmap' )
Say 'The width of the pixmap is' dw_pixmap_width( pixmap )
```

6.1.6.9. Rendering/DW_pixmap_height [Functions]

[Top] [Rendering] [Functions]

NAME

DW_pixmap_height

SYNOPSIS

width = dw_pixmap_height(Win)

FUNCTION

returns the height of the specified pixmap.

ARGUMENTS

- Win - the window identifier returned from DW_pixmap_new(), DW_pixmap_new_from_file() or DW_pixmap_new_from_data().

RESULT

The pixmap's height

SEE ALSO

DW_pixmap_width

SOURCE

```
render = dw_render_new( 10 )
pixmap = dw_pixmap_new_from_file( render, '/home/mark/mypixmap' )
Say 'The height of the pixmap is' dw_pixmap_height( pixmap )
```

6.1.6.10. Rendering/DW_draw_text [Functions]

[Top] [Rendering] [Functions]

NAME

DW_draw_text

SYNOPSIS

dw_draw_text(Win, Pixmap, Fill, X, Y, Text)

FUNCTION

Draws the specified Text at the coordinates X/Y. The colour of the Text is set by a call to DW_color_foreground_set(). The font of the text is set by a call to DW_window_set_font() against the render box in which the text is drawn eith directly, or via a pixmap.

ARGUMENTS

- Win - the window identifier returned from DW_render_new()
- Pixmap - the pixmap identifier returned from DW_pixmap_new()
- X - the x coordinate in pixels of the top left corner
- Y - the y coordinate in pixels of the top left corner
- Text - the text to be drawn

RESULT

No return result

SEE ALSO

DW_render_new, DW_pixmap_new, DW_color_foreground_set DW_draw_line, DW_draw_point, DW_draw_rect, DW_window_set_font

NOTES

Only one of Win or Pixmap is required. The other argument should be set to 0.

SOURCE

```
...
win = dw_render_new( 0 )
Call dw_window_set_font win, myfont
...
Call dw_draw_text win, 0, 10, 34, 'A string at 10/34'
```

6.1.6.11. Rendering/DW_draw_line [Functions]

[Top] [Rendering] [Functions]

NAME

DW_draw_line

SYNOPSIS

dw_draw_line(Win, Pixmap, X1, Y1, X2, Y2)

FUNCTION

Draws a line between the coordinates X1/Y1 and X2/Y2 in either the render window identfied by Win or the pixmap identified by Pixmap. The line is 1 pixel in width and the colour of the line is set by a call to DW_color_foreground_set().

ARGUMENTS

- Win - the window identifier returned from DW_render_new()
- Pixmap - the pixmap identifier returned from DW_pixmap_new()
- X1 - the x coordinate in pixels of one end of the line
- Y1 - the y coordinate in pixels of one end of the line
- X2 - the x coordinate in pixels of the other end of the line
- Y2 - the y coordinate in pixels of the other end of the line

RESULT

No return result

SEE ALSO

DW_render_new, DW_pixmap_new, DW_color_foreground_set, DW_draw_point, DW_draw_rect, DW_draw_text

NOTES

Only one of Win or Pixmap is required. The other argument should be set to 0.

SOURCE

```
...
win = dw_render_new( 0 )
...
Call dw_draw_line win, 0, 10, 10, 34, 56
```

6.1.6.12. Rendering/DW_draw_rect [Functions]

[Top] [Rendering] [Functions]

NAME

DW_draw_rect

SYNOPSIS

dw_draw_rect(Win, Pixmap, Fill, X, Y, Width, Height)

FUNCTION

Draws a rectangle with the top left corner at the coordinates X/Y with the specified Width and Height. All measurements are in pixels. The colour of the rectangle is set by a call to DW_color_foreground_set().

ARGUMENTS

- Win - the window identifier returned from DW_render_new()
- Pixmap - the pixmap identifier returned from DW_pixmap_new()
- Fill - indicate if a filled or outline rectangle is drawn
- X - the x coordinate in pixels of the top left corner
- Y - the y coordinate in pixels of the top left corner
- Width - the width of the rectangle
- Height - the height of the rectangle

RESULT

No return result

SEE ALSO

DW_render_new, DW_pixmap_new, DW_color_foreground_set DW_draw_line, DW_draw_point, DW_draw_text, DW_draw_polygon, DrawingFlags

NOTES

Only one of Win or Pixmap is required. The other argument should be set to 0.

SOURCE

```
...
win = dw_render_new( 0 )
...
Call dw_draw_rect win, 0, !REXXDW.!DW_DRAW_FILL, 10, 34, 40, 40
```

6.1.6.13. Rendering/DW_draw_point [Functions]

[Top] [Rendering] [Functions]

NAME

DW_draw_point

SYNOPSIS

dw_draw_point(Win, Pixmap, X, Y)

FUNCTION

Draws a point 1 pixel in size at the coordinates X/Y. The colour of the point is set by a call to DW_color_foreground_set().

ARGUMENTS

- Win - the window identifier returned from DW_render_new()
- Pixmap - the pixmap identifier returned from DW_pixmap_new()
- X - the x coordinate in pixels of the point
- Y - the y coordinate in pixels of the point

RESULT

No return result

SEE ALSO

DW_render_new, DW_pixmap_new, DW_color_foreground_set DW_draw_line, DW_draw_rect, DW_draw_text

NOTES

Only one of Win or Pixmap is required. The other argument should be set to 0.

SOURCE

```
...
win = dw_render_new( 0 )
...
Call dw_draw_point win, 0, 10, 34
```

6.1.6.14. Rendering/DW_draw_polygon [Functions]

[Top] [Rendering] [Functions]

NAME

DW_draw_polygon

SYNOPSIS

dw_draw_polygon(Win, Pixmap, Fill, X, Y)

FUNCTION

Draws a closed polygon with the points defined in the X and Y stems. All measurements are in pixels. The colour of the polygon is set by a call to DW_color_foreground_set().

ARGUMENTS

- Win - the window identifier returned from DW_render_new()
- Pixmap - the pixmap identifier returned from DW_pixmap_new()
- Fill - indicate if a filled or outline polygon is drawn

- X - a stem name of the x coordinate in pixels
- Y - a stem name of the y coordinate in pixels

RESULT

No return result

SEE ALSO

DW_render_new, DW_pixmap_new, DW_color_foreground_set DW_draw_line, DW_draw_point, DW_draw_text, DW_draw_rect, DrawingFlags

NOTES

Only one of Win or Pixmap is required. The other argument should be set to 0. The ploygon is closed. ie a line is drawn from the first coordinates to the last corrdinates.

SOURCE

```
...
win = dw_render_new( 0 )
...
x.0 = 3
x.1 = 10
x.2 = 20
x.3 = 30
y.0 = 3
y.1 = 15
y.2 = 45
y.3 = 77
Call dw_draw_polygon win, 0, !REXXDW.!DW_DRAW_FILL, 'x.', 'y.'
```

6.1.7. Widgets/EntryField [Modules]

[Top] [Widgets] [Modules]

DESCRIPTION

Entryfields allow a user to enter a single line of text.

6.1.7.1. EntryField/DW_entryfield_new [Functions]

[Top] [EntryField] [Functions]

NAME

DW_entryfield_new

SYNOPSIS

win = dw_entryfield_new(CurrentText, Id)

FUNCTION

Creates a new entryfield. An entryfield is a single line window into which text can be entered.

ARGUMENTS

- CurrentText - the initial text to display in the entryfield
- Id - a numeric identifier used to identify the window

RESULT

A window identifier.

SEE ALSO

DW_entryfield_password_new, DW_entryfield_set_limit

SOURCE

```
...
entryfield = dw_entryfield_new( "Initial value", 0 )
```

6.1.7.2. EntryField/DW_entryfield_password_new [Functions]

[Top] [EntryField] [Functions]

NAME

DW_entryfield_password_new

SYNOPSIS

win = dw_entryfield_password_new(CurrentText, Id)

FUNCTION

Creates a new password entryfield. A password entryfield is similar to an entryfiled, but any character displayed in the entryfield is replaced by an asterisk to mask the contents.

ARGUMENTS

- CurrentText - the initial text to display in the entryfield
- Id - a numeric identifier used to identify the window

RESULT

A window identifier.

SEE ALSO

DW_entryfield_new, DW_entryfield_set_limit

SOURCE

```
...
entryfield = dw_entryfield_password_new( "mypassword", 0 )
```

6.1.7.3. EntryField/DW_entryfield_set_limit [Functions]

[Top] [EntryField] [Functions]

NAME

DW_entryfield_set_limit

SYNOPSIS

dw_entryfield_set_limit(Win, Length)

FUNCTION

Sets the maximum number of characters that can be entered into an entryfield.

ARGUMENTS

- Win - the window identifier returned from DW_entryfield_new() or DW_entryfield_password_new()
- Length - the maximum number of characters allowed

RESULT

No return result.

SEE ALSO

DW_entryfield_new, DW_entryfield_password_new

SOURCE

```
...
entryfield = dw_entryfield_password_new( "mypassword", 0 )
Call dw_entryfield_set_limit entryfield, 10
```

6.1.8. Widgets/ListboxAndCombobox [Modules]

[Top] [Widgets] [Modules]

DESCRIPTION

Listboxes provide a list of selectable items. A combobox is a listbox plus an entry field.

6.1.8.1. ListboxAndCombobox/DW_listbox_new [Functions]

[Top] [ListboxAndCombobox] [Functions]

NAME

DW_listbox_new

SYNOPSIS

win = dw_listbox_new(Id, Selection)

FUNCTION

Creates a new listbox. A listbox contains a list of item values.

ARGUMENTS

- Id - a numeric identifier used to identify the window
- Selection - indicates if one or more items are selectable at once

RESULT

A window identifier.

SEE ALSO

DW_combobox_new, ListboxSelection

SOURCE

```
...
listbox = dw_listbox_new( 10, !REXXDW.!DW_LB_SINGLE_SELECTION )
```

6.1.8.2. ListboxAndCombobox/DW_combobox_new [Functions]

[Top] [ListboxAndCombobox] [Functions]

NAME

DW_combobox_new

SYNOPSIS

win = dw_combobox_new(ComboboxText, Id)

FUNCTION

Creates a new combobox. A combobox contains a single line text entry field plus a selectable item from a list of pre set item values.

ARGUMENTS

- ButtonText - the text to display as the default value of the combobox
- Id - a numeric identifier used to identify the window

RESULT

A window identifier.

SEE ALSO

DW_listbox_new

SOURCE

```
...
combobox = dw_combobox_new( "Default", 0 )
```

6.1.8.3. ListboxAndCombobox/DW_listbox_selector_new [Functions]

[Top] [ListboxAndCombobox] [Functions]

NAME

DW_listbox_selector_new

SYNOPSIS

win = dw_listbox_selector_new(LeftHeading, RightHeading, Width, Height)

FUNCTION

Creates a new Listbox Selector widget. A Listbox Selector widget provides two listboxes side-by-side with buttons to move items from one listbox to another.

ARGUMENTS

- LeftHeading - the text to appear above the left Listbox
- RightHeading - the text to appear above the right Listbox
- Width - the minumum width of the Listboxes
- Height - the minumum height of the Listboxes

RESULT

A window identifier, which is actually a Box widget suitable for packing.

SOURCE

```
ls = dw_listbox_selector_new( 'Selected', 'Available', 100, 200 )
value.0 = 3
value.1 = 'line 1'
value.2 = 'line 2'
value.3 = 'line 3'
ind.0 = 3
ind.1 = 1
ind.2 = 1
ind.3 = 0
Call dw_listbox_selector_setup( ls, 'value.', 'ind.' )
...
Do i = 1 To value.0
   If ind.i = 1 Then Say 'Value:' value.i 'has been selected'
End
```

6.1.8.4. ListboxAndCombobox/DW_listbox_selector_setup [Functions]

[Top] [ListboxAndCombobox] [Functions]

NAME

DW_listbox_selector_setup

SYNOPSIS

dw_listbox_selector_setup(Win, ValueArray, SetArray)

FUNCTION

Populates a listbox selector widget.

ARGUMENTS

- Win - the window identifier returned from DW_listbox_selector_new()
- ValueArray - an array containing the text to be displayed in the Listboxes
- SetArray - an array specifying which values in ValueArray are displayed in which Listbox

RESULT

No return result.

NOTES

The ValueArray is a Rexx "array" containing the items that are to be displayed in the Listboxes. The SetArray is a Rexx "array" with values set to 1 or 0. When creating the Listbox Selector, if an item in the SetArray has

a value of 1 the corresponding item in the ValueArray is displayed in the left Listbox. When an item is moved from the left Listbox to the right Listbox, the corresponding item in the SetArray is set to 1. When an item is moved from the right Listbox to the left Listbox, the corresponding item in the SetArray is set to 0. The ValueArray and SetArray stem variable names should be global, so that they can be referenced inside RexxDW callbacks.

SOURCE

```
ls = dw_listbox_selector_new( 'Selected', 'Available', 100, 200 )
value.0 = 3
value.1 = 'line 1'
value.2 = 'line 2'
value.3 = 'line 3'
ind.0 = 3
ind.1 = 1
ind.2 = 1
ind.3 = 0
Call dw_listbox_selector_setup( ls, 'value.', 'ind.' )
...
Do i = 1 To value.0
   If ind.i = 1 Then Say 'Value:' value.i 'has been selected'
End
```

6.1.8.5. ListboxAndCombobox/DW_listbox_selector_clear [Functions]

[Top] [ListboxAndCombobox] [Functions]

NAME

DW_listbox_selector_clear

SYNOPSIS

dw_listbox_selector_clear(Win)

FUNCTION

Clears all data from a listbox selector widget.

ARGUMENTS

- Win - the window identifier returned from DW_listbox_selector_new()

RESULT

No return result.

SOURCE

```
ls = dw_listbox_selector_new( 'Selected', 'Available', 100, 200 )
value.0 = 3
value.1 = 'line 1'
```

```
value.2 = 'line 2'
value.3 = 'line 3'
ind.0 = 3
ind.1 = 1
ind.2 = 1
ind.3 = 0
Call dw_listbox_selector_setup( ls, 'value.', 'ind.' )
...
Do i = 1 To value.0
   If ind.i = 1 Then Say 'Value:' value.i 'has been selected'
End
...
Call dw_listbox_selector_clear( ls )
```

6.1.8.6. ListboxAndCombobox/DW_listbox_append [Functions]

[Top] [ListboxAndCombobox] [Functions]

NAME

DW_listbox_append

SYNOPSIS

dw_listbox_append(Win, Text)

FUNCTION

Appends the specified Text to the end of the current items in the listbox or combobox specified by Win.

ARGUMENTS

- Win - the window identifier returned from DW_listbox_new() or DW_combobox_new
- Text - the text of the item to append to the list

RESULT

No return result.

SEE ALSO

DW_listbox_list_append, DW_listbox_insert

SOURCE

```
listbox = dw_listbox_new( 0, !REXXDW.!DW_LB_SINGLE_SELECTION )
Do i = 1 To 5
   Call dw_listbox_append listbox, text.i
End
```

6.1.8.7. ListboxAndCombobox/DW_listbox_list_append [Functions]

[Top] [ListboxAndCombobox] [Functions]

NAME

DW_listbox_list_append

SYNOPSIS

dw_listbox_list_append(Win, TextArray)

FUNCTION

Appends each item in the specified TextArray to the end of the current items in the listbox or combobox specified by Win.

ARGUMENTS

- Win - the window identifier returned from DW_listbox_new() or DW_combobox_new
- TextArray - an array containing text of the items to append to the list

RESULT

No return result.

SEE ALSO

DW_listbox_append, DW_listbox_insert

SOURCE

```
listbox = dw_listbox_new( 0, !REXXDW.!DW_LB_SINGLE_SELECTION )
text.0 = 3
text.1 = 'line 1'
text.2 = 'line 2'
text.3 = 'line 3'
Call dw_listbox_list_append listbox, 'text.'
```

6.1.8.8. ListboxAndCombobox/DW_listbox_insert [Functions]

[Top] [ListboxAndCombobox] [Functions]

NAME

DW_listbox_insert

SYNOPSIS

dw_listbox_insert(Win, Text, Pos)

FUNCTION

Appends the specified Text into the listbox or combobox specified by Win at the specified Pos.

ARGUMENTS

- Win - the window identifier returned from DW_listbox_new() or DW_combobox_new
- Text - the text of the item to insert into the list
- Pos - the position at which the text is to be inserted. If zero, negative, or greater than the number of existing items, Text will be appended

RESULT

No return result.

SEE ALSO

DW_listbox_append

SOURCE

```
listbox = dw_listbox_new( 0, !REXXDW.!DW_LB_SINGLE_SELECTION )
Call dw_listbox_insert listbox, 'first', 1
Call dw_listbox_insert listbox, 'last', 1
```

6.1.8.9. ListboxAndCombobox/DW_listbox_clear [Functions]

[Top] [ListboxAndCombobox] [Functions]

NAME

DW_listbox_clear

SYNOPSIS

dw_listbox_clear(Win)

FUNCTION

Removes all items from the listbox or combobox specified by Win.

ARGUMENTS

- Win - the window identifier returned from DW_listbox_new() or DW_combobox_new

RESULT

No return result.

SEE ALSO

DW_listbox_delete

SOURCE

```
listbox = dw_listbox_new( 0, !REXXDW.!DW_LB_SINGLE_SELECTION )
Do i = 1 To 5
   Call dw_listbox_append listbox, text.i
End
...
Call dw_listbox_clear( listbox )
```

6.1.8.10. ListboxAndCombobox/DW_listbox_count [Functions]

[Top] [ListboxAndCombobox] [Functions]

NAME

DW_listbox_count

SYNOPSIS

count = dw_listbox_count(Win)

FUNCTION

Returns the number of items in the listbox or combobox specified by Win.

ARGUMENTS

- Win - the window identifier returned from DW_listbox_new() or DW_combobox_new

RESULT

The number of items in the list.

SOURCE

```
combobox = dw_combobox_new( "Default", 0 )
Do i = 1 To 5
   Call dw_listbox_append combobox, text.i
End
Say 'There are' dw_listbox_count( combobox ) 'items in the list'
```

6.1.8.11. ListboxAndCombobox/DW_listbox_delete [Functions]

[Top] [ListboxAndCombobox] [Functions]

NAME

DW_listbox_delete

SYNOPSIS

dw_listbox_delete(Win, Index)

FUNCTION

Deletes the item specified by Index from the list of items in the listbox or combobox specified by Win.

ARGUMENTS

- Win - the window identifier returned from DW_listbox_new() or DW_combobox_new
- Index - the 0 based index of the item to delete

RESULT

No return result.

SEE ALSO

DW_listbox_clear

SOURCE

```
combobox = dw_combobox_new( "Default", 0 )
Do i = 1 To 5
   Call dw_listbox_append combobox, text.i
End
...
Call dw_listbox_delete combobox, 4
```

6.1.8.12. ListboxAndCombobox/DW_listbox_select [Functions]

[Top] [ListboxAndCombobox] [Functions]

NAME

DW_listbox_select

SYNOPSIS

dw_listbox_select(Win, Index, State)

FUNCTION

Sets the State of the item specified by Index in the list of items in the listbox or combobox specified by Win.

ARGUMENTS

- Win - the window identifier returned from DW_listbox_new() or DW_combobox_new
- Index - the 0 based index of the item to change its state
- State - indicates whether the item is to be selected or not

RESULT

No return result.

SEE ALSO

DW_listbox_selected, ListboxSelected

NOTES

If the listbox was created with DW_LB_SINGLE_SELECTION, then setting an item to be selected will unselect an item that is currently selected. If the listbox was created with DW_LB_MULTIPLE_SELECTION, then setting an item to be selected will leave any other item already selected alone.

SOURCE

```
listbox = dw_listbox_new( 10, !REXXDW.!DW_LB_SINGLE_SELECTION )
Do i = 1 To 5
   Call dw_listbox_append listbox, text.i
End
...
Call dw_listbox_select listbox, 4, !REXXDW.!DW_LB_SELECTED
```

6.1.8.13. ListboxAndCombobox/DW_listbox_selected [Functions]

[Top] [ListboxAndCombobox] [Functions]

NAME

DW_listbox_selected

SYNOPSIS

index = dw_listbox_selected(Win)

FUNCTION

Returns the index of the item in the listbox or combobox currently selected or !REXXDW.!DW_LB_NONE if none selected.

ARGUMENTS

- Win - the window identifier returned from DW_listbox_new() or DW_combobox_new

RESULT

The 0 based index of the item selected.

SEE ALSO

ListboxReturn, DW_listbox_select, DW_listbox_selected_multi

SOURCE

```
listbox = dw_listbox_new( 10, !REXXDW.!DW_LB_SINGLE_SELECTION )
Do i = 1 To 5
   Call dw_listbox_append listbox, text.i
End
...
Call dw_listbox_select listbox, 4, !REXXDW.!DW_LB_SELECTED
...
Say 'Item' dw_listbox_selected( listbox ) 'is currently selected'
```

6.1.8.14. ListboxAndCombobox/DW_listbox_selected_multi [Functions]

[Top] [ListboxAndCombobox] [Functions]

NAME

DW_listbox_selected_multi

SYNOPSIS

index = dw_listbox_selected_multi(Win, Start)

FUNCTION

Returns the index of the next item in the listbox currently selected or !REXXDW.!DW_LB_NONE if none selected.

ARGUMENTS

- Win - the window identifier returned from DW_listbox_new()
- Start - the index to start looking from

RESULT

The 0 based index of the next item selected.

SEE ALSO

ListboxReturn, DW_listbox_select, DW_listbox_selected

NOTES

This call only works on listboxes created with DW_LB_MULTIPLE_SELECTION; and not on comboboxes. The first call should use -1 as Start, and on subsequent calls, the index returned from the previous call.

SOURCE

```
listbox = dw_listbox_new( 10, !REXXDW.!DW_LB_MULTIPLE_SELECTION )
Do i = 1 To 5
   Call dw_listbox_append listbox, text.i
End
...
Call dw_listbox_select listbox, 2, !REXXDW.!DW_LB_SELECTED
Call dw_listbox_select listbox, 4, !REXXDW.!DW_LB_SELECTED
...
idx = -1
Do Forever
   idx = dw_listbox_selected_multi( listbox, idx )
   If idx = !REXXDW.!DW_LB_NONE Then Leave
   Say 'Item' idx 'is currently selected'
End
```

6.1.8.15. ListboxAndCombobox/DW_listbox_set_top [Functions]

[Top] [ListboxAndCombobox] [Functions]

NAME

DW_listbox_set_top

SYNOPSIS

dw_listbox_set_top(Win, Index)

FUNCTION

Sets the item with Index to be the first item displayed in the viewport of the listbox.

ARGUMENTS

- Win - the window identifier returned from DW_listbox_new()
- Index - the 0 based index of the item to display at the top of the viewport

RESULT

No return result.

NOTES

This does not work on comboboxes.

SOURCE

```
listbox = dw_listbox_new( 10, !REXXDW.!DW_LB_SINGLE_SELECTION )
```

```
Do i = 1 To 5
   Call dw_listbox_append listbox, text.i
End
...
Call dw_listbox_set_top listbox, 4
```

6.1.8.16. ListboxAndCombobox/DW_listbox_set_text [Functions]

[Top] [ListboxAndCombobox] [Functions]

NAME

DW_listbox_set_text

SYNOPSIS

dw_listbox_set_text(Win, Index, Text)

FUNCTION

Sets the Text of the Index item in the listbox or combobox

ARGUMENTS

- Win - the window identifier returned from DW_listbox_new() or DW_combobox_new
- Index - the 0 based index of the item to change its text
- Text - the new text of the item

RESULT

No return result.

SEE ALSO

DW_listbox_append

SOURCE

```
listbox = dw_listbox_new( 10, !REXXDW.!DW_LB_SINGLE_SELECTION )
Do i = 1 To 5
   Call dw_listbox_append listbox, text.i
End
...
Call dw_listbox_set_text listbox, 4, 'New Text'
```

6.1.8.17. ListboxAndCombobox/DW_listbox_get_text [Functions]

[Top] [ListboxAndCombobox] [Functions]

NAME

DW_listbox_get_text

SYNOPSIS

text = dw_listbox_get_text(Win, Index)

FUNCTION

Returns the text of the item with Index in the listbox or combobox

ARGUMENTS

- Win - the window identifier returned from DW_listbox_new() or DW_combobox_new
- Index - 0 based index into the list of items for which the text is required

RESULT

The text of the item.

SOURCE

```
combobox = dw_combobox_new( "Default", 0 )
Do i = 1 To 5
   Call dw_listbox_append combobox, text.i
End
Say 'Item 3 is' dw_listbox_get_text( combobox, 3 )
```

6.1.9. Widgets/EmbeddedHTML [Modules]

[Top] [Widgets] [Modules]

DESCRIPTION

The embedded HTML widget provides a means to display HTML in a window.

6.1.9.1. EmbeddedHTML/DW_html_new [Functions]

[Top] [EmbeddedHTML] [Functions]

NAME

DW_html_new

SYNOPSIS

win = dw_html_new(Id)

FUNCTION

Creates a new HTML widget.

ARGUMENTS

- Id - a numeric identifier used to identify the window

RESULT

A window identifier.

SOURCE

```
...
html = dw_html_new( 10 )
```

6.1.9.2. EmbeddedHTML/DW_html_url [Functions]

[Top] [EmbeddedHTML] [Functions]

NAME

DW_html_url

SYNOPSIS

rc = dw_html_url(Win, Url)

FUNCTION

Displays a URL in a HTML widget.

ARGUMENTS

- Win - the window identifier returned from DW_html_new()
- Url - the URL of the HTML page to display

RESULT

0 if successful; any other value is an error

SOURCE

```
...
html = dw_html_new( 10 )
Call dw_html_url( html, 'http://rexxdw.sf.net' )
```

6.1.9.3. EmbeddedHTML/DW_html_raw [Functions]

[Top] [EmbeddedHTML] [Functions]

NAME

DW_html_raw

SYNOPSIS

rc = dw_html_raw(Win, Raw)

FUNCTION

DIsplays raw HTML in a HTML widget.

ARGUMENTS

- Win - the window identifier returned from DW_html_new()
- Raw - the raw HTML

RESULT

0 if successful; any other value is an error

SOURCE

```
...
html = dw_html_new( 10 )
Call dw_html_raw( html, '<html><body><h1>Rexx/DW rules</h1></body></html>' )
```

6.1.9.4. EmbeddedHTML/DW_html_action [Functions]

[Top] [EmbeddedHTML] [Functions]

NAME

DW_html_action

SYNOPSIS

rc = dw_html_action(Win, Action

FUNCTION

Invoke specified action on HTML widget.

ARGUMENTS

- Win - the window identifier returned from DW_html_new()
- Action - the action to execute. see HTMLActions

RESULT

0 if successful; any other value is an error

SEE ALSO

HTMLActions

SOURCE

```
...
html = dw_html_new( 10 )
Call dw_html_action( html, !REXXDW.!DW_HTML_GOHOME )
```

6.1.10. Widgets/Menu [Modules]

[Top] [Widgets] [Modules]

DESCRIPTION

Menus provide a list of textual options to the user to select an action. RexxDW provides standard drop-down menus located directly under the window's titlebar, or floating popup menus.

6.1.10.1. Menu/DW_menu_new [Functions]

[Top] [Menu] [Functions]

NAME

DW_menu_new

SYNOPSIS

win = dw_menu_new(Id)

FUNCTION

Creates a new menu. This menu can be used as a submenu linked to a menubar, or as a popup menu.

ARGUMENTS

- Id - a numeric identifier used to identify the window

RESULT

A window identifier.

SEE ALSO

DW_menubar_new

SOURCE

```
...
menu = dw_menu_new( 10 )
```

6.1.10.2. Menu/DW_menubar_new [Functions]

[Top] [Menu] [Functions]

NAME

DW_menubar_new

SYNOPSIS

win = dw_menubar_new(Win)

FUNCTION

Creates a new menubar. A menubar is located at the top of the specified toplevel window identified by Win.

ARGUMENTS

- Win - a window identifier returned from a call to DW_window_new().

RESULT

A menubar identifier.

SEE ALSO

DW_menu_new, DW_window_new

NOTES

Currently under GTK+ you need to call this function AFTER other widgets have been created in the window that this menubar is to reside.

SOURCE

```
...
win = dw_window_new( !REXXDW.!DW_DESKTOP, 'Window on desktop', style )
```

```
...
menubar = dw_menubar_new( win )
```

6.1.10.3. Menu/DW_menu_append_item [Functions]

[Top] [Menu] [Functions]

NAME

DW_menu_append_item

SYNOPSIS

index = dw_menu_append_item(Menu, Title, Id, Flags, End, Check, Submenu)

FUNCTION

Append a menu item to an existing menu or menubar. The menu item can be an individual item or another submenu.

ARGUMENTS

- Menu - the menu identifier returned from DW_menu_new() or DW_menubar_new()
- Title - the text of the menu item or DW_MENU_SEPARATOR
- Id - a numeric identifier used to identify the menu item or one of DW_MENU_POPUP, DW_MENU_AUTO
- Flags - effectively ignored
- End - indicates if the menu item appears at the start or end of the existing list. One of DW_MENU_START or DW_MENU_END
- Check - indicates if the menu item is checkable or not via DW_menu_item_set_state() one of DW_MENU_CHECKABLE or DW_MENU_NOT_CHECKABLE
- Submenu - a menu identifier returned from a a call to DW_menu_new() or DW_MENU_NOMENU if this is NOT a sub-menu

RESULT

A menu identifier.

SEE ALSO

DW_menu_new, DW_menubar_new, DW_menu_item_set_state, MenuConstants

NOTES

Preceding a letter of the Title, will enable that letter to be typed on the keyboard to invoke the callback associated with that menu item. In the source below, with the menu item displayed, typing 'a' will execute 'AddRepositoryCallback' The Title can also be set to DW_MENU_SEPARATOR which will display a line instead of text.

A menu item MUST have a unique Id within a particular menu, otherwise the callbacks will not work.

SOURCE

```
menu = dw_menu_new( 0 )
menuitem = dw_menu_append_item( menu, '~Add Repository', 1011, 0, ,
   !REXXDW.!DW_MENU_END, !REXXDW.!DW_MENU_CHECKABLE, !REXXDW.!DW_MENU_NOMENU0 )
Call dw_signal_connect menuitem, !REXXDW.!DW_CLICKED_EVENT, ,
   'AddRepositoryCallback'
...
Call dw_menu_item_set_state menu, 1011, !REXXDW.!DW_CHECKED
```

6.1.10.4. Menu/DW_menu_delete_item [Functions]

[Top] [Menu] [Functions]

NAME

DW_menu_delete_item

SYNOPSIS

dw_menu_delete_item(Menu, Id)

FUNCTION

Deletes a menu item from an existing menu or menubar. The menu item can be an individual item or another submenu.

ARGUMENTS

- Menu - the menu identifier returned from DW_menu_new() or DW_menubar_new()
- Id - a numeric identifier used to identify the menu item

RESULT

No return result.

SEE ALSO

DW_menu_new, DW_menubar_new, DW_menu_append_item, MenuConstants

SOURCE

```
menu = dw_menu_new( 0 )
menuitem = dw_menu_delete_item( menu, '~Add Repository', 1011, 0, ,
   !REXXDW.!DW_MENU_END, !REXXDW.!DW_MENU_NOT_CHECKABLE, 0 )
Call dw_signal_connect menuitem, !REXXDW.!DW_CLICKED_EVENT, ,
   'AddRepositoryCallback'
...
Call dw_menu_delete_item, menu, 1011
```

6.1.10.5. Menu/DW_menu_destroy [Functions]

[Top] [Menu] [Functions]

NAME

DW_menu_destroy

SYNOPSIS

dw_menu_destroy(Menu)

FUNCTION

Destroys a menu or menubar.

ARGUMENTS

- Menu - the menu identifier returned from DW_menu_new() or DW_menubar_new()

RESULT

No return result.

SEE ALSO

DW_menu_new, DW_menubar_new

SOURCE

```
menu = dw_menu_new( 0 )
...
Call dw_menu_destroy menu
...
```

6.1.10.6. Menu/DW_menu_item_set_state [Functions]

[Top] [Menu] [Functions]

NAME

DW_menu_item_set_state

SYNOPSIS

dw_menu_item_set_state(Menu, Id, State)

FUNCTION

Sets the state of the menu item identified by Menu and Id. The state of a menu item can be checked/unchecked and/or enabled/disabled.

ARGUMENTS

- Menu - the menu identifier returned from DW_menu_new() or DW_menubar_new()
- Id - a numeric identifier used to identify the menu
- State - the state of the menu item; the checked state ORed with the enabled state

RESULT

No return result.

SEE ALSO

DW_menu_new, MenuItemConstants

NOTES

The State is cumulative. Just setting DW_CHECKED does not affect whether the menu item is enabled or disabled.

SOURCE

```
menu = dw_menu_new( 0 )
menuitem = dw_menu_append_item( menu, '~Add Repository', 1011, 0, ,
   !REXXDW.!DW_MENU_END, !REXXDW.!DW_MENU_CHECKABLE, 0 )
...
Call dw_menu_item_set_state menu, 1011, dw_or( !REXXDW.!DW_MENU_CHECKED, !REXXDW.!DW_MENU_ENABLED
```

6.1.10.7. Menu/DW_menu_popup [Functions]

[Top] [Menu] [Functions]

NAME

DW_menu_popup

SYNOPSIS

dw_menu_popup(Menu, Win, X, Y)

FUNCTION

Displays the popup Menu in the specified window at the location specified by X/Y in pixels relative to ?????.

ARGUMENTS

- Menu - the menu identifier returned from DW_menu_new()
- Win - the window identifier of the toplevel window
- X - the X coordinate where the top left corner of the menu will appear
- Y - the Y coordinate where the top left corner of the menu will appear

RESULT

No return result.

SEE ALSO

DW_menu_new

NOTES

The popup menu associated with the Menu identifier is destroyed each time **DW_menu_popup**() is called. The Win argument must be the window identifier of the topelevel window. ie. The window identifier returned from DW_window_new() called with !REXXDW.!DW_DESKTOP.

SOURCE

```
win = dw_window_new( !REXXDW.!DW_DESKTOP, 'Window on desktop', style )
...
menu = dw_menu_new( 0 )
menuitem = dw_menu_append_item( menu, '~Add Repository', 1011, 0, ,
   !REXXDW.!DW_MENU_END, !REXXDW.!DW_MENU_NOT_CHECKABLE, 0 )
...
Call dw_menu_popup menu, win, 30, 130
```

6.1.11. Widgets/MultiLineEdit [Modules]

[Top] [Widgets] [Modules]

DESCRIPTION

A multi-line edit widget is similar to an Entryfield, except that more than one line of text can be manipulated.

6.1.11.1. MultiLineEdit/DW_mle_new [Functions]

[Top] [MultiLineEdit] [Functions]

NAME

DW_mle_new

SYNOPSIS

win = dw_mle_new(Id)

FUNCTION

Creates a new Multi-Line edit widget. An MLE is used to display and edit multiple lines of text.

ARGUMENTS

- Id - a numeric identifier used to identify the window

RESULT

An MLE identifier.

SOURCE

```
mle = dw_mle_new( 0 )
```

6.1.11.2. MultiLineEdit/DW_mle_delete [Functions]

[Top] [MultiLineEdit] [Functions]

NAME

DW_mle_delete

SYNOPSIS

dw_mle_delete(Win[, StartPoint, Length])

FUNCTION

Deletes the text from a Multi-Line Entry widget starting at StartPoint for Length characters

ARGUMENTS

- Win - the window identifier returned from DW_mle_new()
- StartPoint - 0 based offset into the MLE counted in characters
- Length - the number of characters to delete

RESULT

No return result.

NOTES

If no Startpoint is specified, it defaults to 0 (the start of the MLE) If no Length is specified, it defaults to the total number of characters in the MLE.

SOURCE

```
...
mle = dw_mle_new( 0 )
...
Call dw_mle_delete( mle, 0, 100 )
```

6.1.11.3. MultiLineEdit/DW_mle_import [Functions]

[Top] [MultiLineEdit] [Functions]

NAME

DW_mle_import

SYNOPSIS

Location = dw_mle_import(Win, Text, StartPoint)

FUNCTION

Inserts the Text into a Multi-Line Entry widget starting at StartPoint.

ARGUMENTS

- Win - the window identifier returned from DW_mle_new()
- Text - the text to insert into the MLE
- StartPoint - 0 based offset into the MLE counted in characters

RESULT

The point into the MLE after the last character in Text was inserted counted in characters.

SEE ALSO

DW_mle_export

NOTES

You need to append a CRLF ('0d0a'x) to the end of text to insert lines. The first call to dw_mle_import() must specify -1 as StartPoint.

SOURCE

```
...
mle = dw_mle_new( 0 )
...
eol = '0d0a'x
loc = -1
Do i = 1 To 10
   loc = dw_mle_import( mle, 'Inserting line' i 'here'||eol, loc )
End
...
```

6.1.11.4. MultiLineEdit/DW_mle_export [Functions]

[Top] [MultiLineEdit] [Functions]

NAME

DW_mle_export

SYNOPSIS

text = dw_mle_export(Win[, StartPoint, Length])

FUNCTION

Returns the text from a Multi-Line Entry widget starting at StartPoint for Length characters.

ARGUMENTS

- Win - the window identifier returned from DW_mle_new()
- StartPoint - 0 based offset into the MLE counted in characters
- Length - the number of characters to export

RESULT

The text exported.

SEE ALSO

DW_mle_import

NOTES

If no Startpoint is specified, it defaults to 0 (the start of the MLE) If no Length is specified, it defaults to the total number of characters in the MLE.

SOURCE

```
...
mle = dw_mle_new( 0 )
...
contents = dw_mle_export( mle, 0, 100 )
```

6.1.11.5. MultiLineEdit/DW_mle_get_size [Functions]

[Top] [MultiLineEdit] [Functions]

NAME

DW_mle_get_size

SYNOPSIS

Bytes Lines = dw_mle_get_size(Win)

FUNCTION

Returns the number of Bytes and Lines currently in the Multi-Line Entry widget.

ARGUMENTS

- Win - the window identifier returned from DW_mle_new()

RESULT

Bytes and Lines are returned as two words. PARSE VALUE is the easiest way to obtain both values.

SEE ALSO

DW_mle_new

SOURCE

```
...
mle = dw_mle_new( 0 )
...
Parse Value dw_mle_get_size( mle ) With bytes lines
```

6.1.11.6. MultiLineEdit/DW_mle_search [Functions]

[Top] [MultiLineEdit] [Functions]

NAME

DW_mle_search

SYNOPSIS

Location = dw_mle_search(Win, Text, StartPoint, SearchFlags)

FUNCTION

Searches the MLE for Text starting at StartPoint.

ARGUMENTS

- Win - the window identifier returned from DW_mle_new()
- Text - the text to search for in the MLE

- StartPoint - 0 based offset into the MLE counted in characters
- SearchFlags- optional flag to search respecting case if set to DW_MLE_CASESENSITIVE. default is to search irrespective of case

RESULT

The point into the MLE where the matching Text starts counted in characters.

SEE ALSO

MLESearchFlags

SOURCE

```
...
mle = dw_mle_new( 0 )
...
loc = dw_mle_import( mle, 'More text here', 100 )
...
pos = dw_mle_search( mle, 'text', 0, 0 )
```

6.1.11.7. MultiLineEdit/DW_mle_set_cursor [Functions]

[Top] [MultiLineEdit] [Functions]

NAME

DW_mle_set_curesor

SYNOPSIS

dw_mle_set_cursor(Win, Point)

FUNCTION

Sets the cursor at Point in the Multi-Line Entry widget.

ARGUMENTS

- Win - the window identifier returned from DW_mle_new()
- Point - the character at which to display the cursor in the MLE

RESULT

No return result.

SOURCE

```
...
mle = dw_mle_new( 0 )
```

```
...
Call dw_mle_set_cursor( mle, 10 )
```

6.1.11.8. MultiLineEdit/DW_mle_set_editable [Functions]

[Top] [MultiLineEdit] [Functions]

NAME

DW_mle_set_editable

SYNOPSIS

dw_mle_set_editable(Win, EditableFlags)

FUNCTION

Sets the state of the Multi-Line Entry widget to be editable or readonly.

ARGUMENTS

- Win - the window identifier returned from DW_mle_new()
- EditableFlags - one of DW_EDITABLE or DW_READONLY

RESULT

No return result.

SEE ALSO

MLEEditableFlags

NOTES

By default MLEs are editable

SOURCE

```
...
mle = dw_mle_new( 0 )
...
Call dw_mle_set_editable( mle, !REXXDW.!DW_READONLY )
```

6.1.11.9. MultiLineEdit/DW_mle_set_visible [Functions]

[Top] [MultiLineEdit] [Functions]

NAME

DW_mle_set_visible

SYNOPSIS

dw_mle_set_visible(Win, Line)

FUNCTION

Sets the Line in the Multi-Line Entry widget to be visible.

ARGUMENTS

- Win - the window identifier returned from DW_mle_new()
- Line - the Line in the MLE to be visible

RESULT

No return result.

SOURCE

```
...
mle = dw_mle_new( 0 )
...
Call dw_mle_set_visible( mle, 10 )
```

6.1.11.10. MultiLineEdit/DW_mle_set_word_wrap [Functions]

[Top] [MultiLineEdit] [Functions]

NAME

DW_mle_set_word_wrap

SYNOPSIS

dw_mle_set_editable(Win, WordWrapFlags)

FUNCTION

Sets the state of the Multi-Line Entry widget to allow lines to wrap on word boundaries or be truncated at the edge of the viewport.

ARGUMENTS

- Win - the window identifier returned from DW_mle_new()
- WordWrapFlags - one of DW_WORD_WRAP or DW_DONT_WORD_WRAP

RESULT

No return result.

SEE ALSO

MLEWordWrapFlags

SOURCE

```
...
mle = dw_mle_new( 0 )
...
Call dw_mle_set_word_wrap( mle, !REXXDW.!DW_WORD_WRAP )
```

6.1.11.11. MultiLineEdit/DW_mle_freeze [Functions]

[Top] [MultiLineEdit] [Functions]

NAME

DW_mle_freeze

SYNOPSIS

dw_mle_freeze(Win)

FUNCTION

Stops the screen output to a Multi-Line Entry widget.

ARGUMENTS

- Win - the window identifier returned from DW_mle_new()

RESULT

No return result.

SEE ALSO

DW_mle_thaw

NOTES

On some platforms, there can be excessive scrolling and redraws when text is imported into an MLE. This function reduces these effects.

SOURCE

```
...
mle = dw_mle_new( 0 )
```

```
...
Call dw_mle_freeze( mle )
loc = -1
Do i = 1 To text.0
   loc = dw_mle_import( mle, text.i, loc )
End
Call dw_mle_thaw( mle )
```

6.1.11.12. MultiLineEdit/DW_mle_thaw [Functions]

[Top] [MultiLineEdit] [Functions]

NAME

DW_mle_thaw

SYNOPSIS

dw_mle_thaw(Win)

FUNCTION

Restarts the screen output to a Multi-Line Entry widget.

ARGUMENTS

- Win - the window identifier returned from DW_mle_new()

RESULT

No return result.

SEE ALSO

DW_mle_freeze

NOTES

On some platforms, there can be excessive scrolling and redraws when text is imported into an MLE. This function reduces these effects.

SOURCE

```
...
mle = dw_mle_new( 0 )
...
Call dw_mle_freeze( mle )
loc = -1
Do i = 1 To text.0
   loc = dw_mle_import( mle, text.i, loc )
End
Call dw_mle_thaw( mle )
```

6.1.12. Widgets/Notebook [Modules]

[Top] [Widgets] [Modules]

DESCRIPTION

Strictly speaking these widgets are "tabbed notebooks". This widget provides another sort of frame in which other widgets are packed. **Notebook** pages are frames that overlap the same screen space, but can be brought to the front by the user clicking on the tab.

6.1.12.1. Notebook/DW_notebook_new [Functions]

[Top] [Notebook] [Functions]

NAME

DW_notebook_new

SYNOPSIS

win = dw_notebook_new(Id, TabLocation)

FUNCTION

Creates a new tabbed notebook widget. A notebook is a frame in which pages containing other widgets can be packed. Each page can be selected by clicking that page's tab.

ARGUMENTS

- Id - a numeric identifier used to identify the window
- TabLocation- indicates if the tabs appear across the top or bottom

RESULT

A notebook identifier.

SEE ALSO

NotebookTabLocation

SOURCE

```
notebook = dw_notebook_new( 0, !REXXDW.!DW_TAB_TO_TOP )
```

6.1.12.2. Notebook/DW_notebook_page_new [Functions]

[Top] [Notebook] [Functions]

NAME

DW_notebook_page_new

SYNOPSIS

page = dw_notebook_page_new(Win, Flags, PageLocation)

FUNCTION

Creates a new page within a tabbed notebook widget.

ARGUMENTS

- Win - the window identifier returned from DW_notebook_new()
- Flags - ignored
- PageLocation - indicates if the new page is shown in front or behind current tabs

RESULT

A notebook page identifier.

SEE ALSO

DW_notebook_new, NotebookPageLocation

NOTES

A maximum of 256 pages can be created in a notebook. On Windows, the PageLocation does nothing, so it is best to specify which page is in front once all pages have been created by calling DW_notebook_page_set().

SOURCE

```
notebook = dw_notebook_new( 0, !REXXDW.!DW_TAB_TO_TOP )
notebookbox = dw_box_new( !REXXDW.!DW_VERT, 0 )
notebookpage = dw_notebook_page_new( notebook, 0, !REXXDW.!DW_PAGE_TO_BACK )
Call dw_notebook_pack notebook, notebookpage, notebookpagebox
```

6.1.12.3. Notebook/DW_notebook_pack [Functions]

[Top] [Notebook] [Functions]

NAME

DW_notebook_pack

SYNOPSIS

dw_notebook_pack(Win, Page, Box)

FUNCTION

Packs a generic box into a page in the notebook.

ARGUMENTS

- Win - the window identifier returned from DW_notebook_new()
- Page - the page identifier returned from DW_notebook_page_new()
- Box - the box identifier returned from a DW_box_new() or DW_groupbox_new()

RESULT

No return result.

SEE ALSO

DW_notebook_new, DW_notebook_page_new

SOURCE

```
notebook = dw_notebook_new( 0, !REXXDW.!DW_TAB_TO_TOP )
notebookbox = dw_box_new( !REXXDW.!DW_VERT, 0 )
notebookpage = dw_notebook_page_new( notebook, 0, !REXXDW.!DW_PAGE_TO_BACK )
Call dw_notebook_pack notebook, notebookpage, notebookpagebox
```

6.1.12.4. Notebook/DW_notebook_page_destroy [Functions]

[Top] [Notebook] [Functions]

NAME

DW_notebook_page_destroy

SYNOPSIS

dw_notebook_page_destroy(Win, Page)

FUNCTION

Destroys the specified Page of a notebook. All widgets inside the Page are also destroyed.

ARGUMENTS

- Win - the window identifier returned from DW_notebook_new()
- Page - the page identifier returned from DW_notebook_page_new()

RESULT

No return result.

SEE ALSO

DW_notebook_new, DW_notebook_page_new

SOURCE

```
notebook = dw_notebook_new( 0, !REXXDW.!DW_TAB_TO_TOP )
notebookbox = dw_box_new( !REXXDW.!DW_VERT, 0 )
notebookpage = dw_notebook_page_new( notebook, 0, !REXXDW.!DW_PAGE_TO_BACK )
Call dw_notebook_pack notebook, notebookpage, notebookpagebox
...
Call dw_notebook_page_destoy notebook, notebookpage
```

6.1.12.5. Notebook/DW_notebook_page_set [Functions]

[Top] [Notebook] [Functions]

NAME

DW_notebook_page_set

SYNOPSIS

dw_notebook_page_set(Win, Page)

FUNCTION

Sets Page to be the notebook page displayed at the front.

ARGUMENTS

- Win - the window identifier returned from DW_notebook_new()
- Page - the page identifier returned from DW_notebook_page_new()

RESULT

No return result.

SEE ALSO

DW_notebook_new, DW_notebook_page_get

SOURCE

```
notebook = dw_notebook_new( 0, !REXXDW.!DW_TAB_TO_TOP )
notebookbox = dw_box_new( !REXXDW.!DW_VERT, 0 )
```

```
notebookpage = dw_notebook_page_new( notebook, 0, !REXXDW.!DW_PAGE_TO_BACK )
Call dw_notebook_pack notebook, notebookpage, notebookpagebox
...
Call dw_notebook_page_set notebook, notebookpage
```

6.1.12.6. Notebook/DW_notebook_page_set_text [Functions]

[Top] [Notebook] [Functions]

NAME

DW_notebook_page_set_text

SYNOPSIS

dw_notebook_page_set_text(Win, Page, Text)

FUNCTION

Sets the text on the notebook tab on the Page to Text.

ARGUMENTS

- Win - the window identifier returned from DW_notebook_new()
- Page - the page identifier returned from DW_notebook_page_new()
- Text - the text to display on the tab

RESULT

No return result.

SOURCE

```
notebook = dw_notebook_new( 0, !REXXDW.!DW_TAB_TO_TOP )
notebookbox = dw_box_new( !REXXDW.!DW_VERT, 0 )
notebookpage = dw_notebook_page_new( notebook, 0, !REXXDW.!DW_PAGE_TO_BACK )
Call dw_notebook_pack notebook, notebookpage, notebookpagebox
...
Call dw_notebook_page_set_text notebook, notebookpage, 'Tab Text'
```

6.1.12.7. Notebook/DW_notebook_page_set_status_text [Functions]

[Top] [Notebook] [Functions]

NAME

DW_notebook_page_set_status_text

SYNOPSIS

dw_notebook_page_set_status_text(Win, Page, Text)

FUNCTION

Sets the status text area on the Page to Text.

ARGUMENTS

- Win - the window identifier returned from DW_notebook_new()
- Page - the page identifier returned from DW_notebook_page_new()
- Text - the text to display in the status text area

RESULT

No return result.

NOTES

Only OS/2 has a status text area. Other platforms ignore this function.

SOURCE

```
notebook = dw_notebook_new( 0, !REXXDW.!DW_TAB_TO_TOP )
notebookbox = dw_box_new( !REXXDW.!DW_VERT, 0 )
notebookpage = dw_notebook_page_new( notebook, 0, !REXXDW.!DW_PAGE_TO_BACK )
Call dw_notebook_pack notebook, notebookpage, notebookpagebox
...
Call dw_notebook_page_set_status_text notebook, notebookpage, 'Subtitle Text'
```

6.1.12.8. Notebook/DW_notebook_page_get [Functions]

[Top] [Notebook] [Functions]

NAME

DW_notebook_page_get

SYNOPSIS

page = dw_notebook_page_get(Win, Flags, PageLocation)

FUNCTION

Return the page identifier of the notebook page currently at the front.

ARGUMENTS

- Win - the window identifier returned from DW_notebook_new()

RESULT

A notebook page identifier.

SEE ALSO

DW_notebook_new, DW_notebook_page_set

SOURCE

```
notebook = dw_notebook_new( 0, !REXXDW.!DW_TAB_TO_TOP )
notebookbox = dw_box_new( !REXXDW.!DW_VERT, 0 )
notebookpage = dw_notebook_page_new( notebook, 0, !REXXDW.!DW_PAGE_TO_BACK )
Call dw_notebook_pack notebook, notebookpage, notebookpagebox
...
Say 'The pageid of the notebook is:' dw_notebook_page_get( notebook )
```

6.1.13. Widgets/Percent [Modules]

[Top] [Widgets] [Modules]

DESCRIPTION

A **Percent** widget is simply a progress bar.

NOTES

Under OS/2 only 1 instance of a percent or slider widget can exist within the same box. So for complete portability, all percent and slider widgets should be enclosed in their own box.

6.1.13.1. Percent/DW_percent_new [Functions]

[Top] [Percent] [Functions]

NAME

DW_percent_new

SYNOPSIS

win = dw_percent_new(Id)

FUNCTION

Creates a new percent progress status bar widget.

ARGUMENTS

- Id - a numeric identifier used to identify the window

RESULT

A percent progress status bar identifier.

SOURCE

```
percent = dw_percent_new( 0 )
```

6.1.13.2. Percent/DW_percent_set_pos [Functions]

[Top] [Percent] [Functions]

NAME

DW_percent_set_pos

SYNOPSIS

dw_percent_set_pos(Win, Position)

FUNCTION

Sets the current position of the percent bar to Position.

ARGUMENTS

- Win - the window identifier returned from DW_percent_new()
- Position - the position of the bar as a whole number percentage or !REXXDW.!DW_PERCENT_INDETERMINATE to display an infinitely moving bar not relative to a fixed value

RESULT

No return result.

SOURCE

```
percent = dw_percent_new( 0 )
...
now = (bytes_sent*100) % total_bytes
Call dw_percent_set_pos percent, now
```

6.1.14. Widgets/Scrollbar [Modules]

[Top] [Widgets] [Modules]

DESCRIPTION

A **Scrollbar** widget is used in conjunction with other widgets that are not by default scrollable, to allow details that are not completely visible in a window to be viewable by scrolling. Both horizontal and vertical scrollbars are supported.

6.1.14.1. Scrollbar/DW_scrollbar_new [Functions]

[Top] [Scrollbar] [Functions]

NAME

DW_scrollbar_new

SYNOPSIS

win = dw_scrollbar_new(Orientation, Id)

FUNCTION

Creates a new scrollbar. A scrollbar is used to scroll various other widgets like Multi-Line edit windows or listboxes.

ARGUMENTS

- Orientation- !REXXDW.!DW_VERT to indicate a vertical scrollbar !REXXDW.!DW_HORZ to indicate a horizontal scrollbar
- Id - a numeric identifier used to identify the window

RESULT

A scrollbar identifier.

SOURCE

```
vscrollbar = dw_scrollbar_new( !REXXDW.!DW_VERT, 0 )
```

6.1.14.2. Scrollbar/DW_scrollbar_set_range [Functions]

[Top] [Scrollbar] [Functions]

NAME

DW_scrollbar_set_range

SYNOPSIS

dw_scrollbar_set_range(Win, Range, Thumbwidth)

FUNCTION

Sets the maximum value and width of the scrollbar thumb of the scrollbar.

ARGUMENTS

- Win - the window identifier returned from DW_scrollbar_new()
- Range - the maximum range of the scrollbar
- Thumbwidth - the width of the scrollbar's thumb

RESULT

No return result.

SOURCE

```
vscrollbar = dw_scrollbar_new( !REXXDW.!DW_VERT, 0 )
Call dw_scrollbar_set_range vscrollbar, 200, 40
...
Call dw_scrollbar_set_pos vscrollbar, 150
...
Say 'The scrollbar is at' dw_scrollbar_get_pos( vscrollbar )
```

6.1.14.3. Scrollbar/DW_scrollbar_set_pos [Functions]

[Top] [Scrollbar] [Functions]

NAME

DW_scrollbar_set_pos

SYNOPSIS

dw_scrollbar_set_pos(Win, Position)

FUNCTION

Sets the position of the scrollbar.

ARGUMENTS

- Win - the window identifier returned from DW_scrollbar_new()
- Position - the position within the scrollbar's range

RESULT

No return result.

SEE ALSO

DW_scrollbar_get_pos

SOURCE

```
vscrollbar = dw_scrollbar_new( !REXXDW.!DW_VERT, 0 )
Call dw_scrollbar_set_range vscrollbar, 200, 40
...
Call dw_scrollbar_set_pos vscrollbar, 150
...
Say 'The scrollbar is at' dw_scrollbar_get_pos( vscrollbar )
```

6.1.14.4. Scrollbar/DW_scrollbar_get_pos [Functions]

[Top] [Scrollbar] [Functions]

NAME

DW_scrollbar_get_pos

SYNOPSIS

pos = dw_scrollbar_get_pos(Win)

FUNCTION

Returns the current position of the scrollbar.

ARGUMENTS

- Win - the window identifier returned from DW_scrollbar_new()

RESULT

The position of the scrollbar.

SEE ALSO

DW_scrollbar_set_pos

SOURCE

```
vscrollbar = dw_scrollbar_new( !REXXDW.!DW_VERT, 0 )
Call dw_scrollbar_set_range vscrollbar, 200, 40
Say 'The scrollbar is at' dw_scrollbar_get_pos( vscrollbar )
```

6.1.15. Widgets/Slider [Modules]

[Top] [Widgets] [Modules]

DESCRIPTION

A **Slider** widget is similar to a Scrollbar, but it allows the user to select a numeric value by dragging a thumb until the desired numeric value is reached.

NOTES

Under OS/2 only 1 instance of a percent or slider widget can exist within the same box. So for complete portability, all percent and slider widgets should be enclosed in their own box.

6.1.15.1. Slider/DW_slider_new [Functions]

[Top] [Slider] [Functions]

NAME

DW_slider_new

SYNOPSIS

win = dw_slider_new(Orientation, Range, Id)

FUNCTION

Creates a new slider. A slider provides a widget for selecting a numeric values from a range.

ARGUMENTS

- Orientation- !REXXDW.!DW_VERT to indicate a vertical scrollbar !REXXDW.!DW_HORZ to indicate a horizontal scrollbar
- Range - the maximum number of the numeric range
- Id - a numeric identifier used to identify the window

RESULT

A slider identifier.

SOURCE

```
slider = dw_slider_new( !REXXDW.!DW_VERT, 100, 0 )
```

6.1.15.2. Slider/DW_slider_set_pos [Functions]

[Top] [Slider] [Functions]

NAME

DW_slider_set_pos

SYNOPSIS

dw_slider_set_pos(Win, Position)

FUNCTION

Sets the position of the slider.

ARGUMENTS

- Win - the window identifier returned from DW_slider_new()
- Position - the position within the slider's range

RESULT

No return result.

SEE ALSO

DW_slider_get_pos

SOURCE

```
vslider = dw_slider_new( !REXXDW.!DW_VERT, 100, 0 )
...
Call dw_slider_set_pos vslider, 50
...
Say 'The slider is at' dw_slider_get_pos( vslider )
```

6.1.15.3. Slider/DW_slider_get_pos [Functions]

[Top] [Slider] [Functions]

NAME

DW_slider_get_pos

SYNOPSIS

pos = dw_slider_get_pos(Win)

FUNCTION

Returns the current position of the slider.

ARGUMENTS

- Win - the window identifier returned from DW_slider_new()

RESULT

The position of the slider.

SEE ALSO

DW_slider_set_pos

SOURCE

```
vslider = dw_slider_new( !REXXDW.!DW_VERT, 100, 0 )
Say 'The slider is at' dw_slider_get_pos( vslider )
```

6.1.16. Widgets/Spinbutton [Modules]

[Top] [Widgets] [Modules]

DESCRIPTION

A **Spinbutton** widget is similar to a Slider, but the user selects a numeric value by clicking an up or down arrow, or by manually entering the value into an entry field. A **Spinbutton** will wrap around at each end of its value limits.

6.1.16.1. Spinbutton/DW_spinbutton_new [Functions]

[Top] [Spinbutton] [Functions]

NAME

DW_spinbutton_new

SYNOPSIS

win = dw_spinbutton_new(Text, Id)

FUNCTION

Creates a new spinbutton. A spinbutton provides a widget for selecting a numeric values from a range or by manually entering a value.

ARGUMENTS

- Text - the text describing the spinbutton
- Id - a numeric identifier used to identify the window

RESULT

A spinbutton identifier.

SOURCE

```
spinbutton = dw_spinbutton_new( 'Select Number', 0 )
```

6.1.16.2. Spinbutton/DW_spinbutton_set_limits [Functions]

[Top] [Spinbutton] [Functions]

NAME

DW_spinbutton_set_limits

SYNOPSIS

dw_spinbutton_set_limits(Win, Upper, Lower)

FUNCTION

Sets the maximum and minimum values the spinbutton can support.

ARGUMENTS

- Win - the window identifier returned from DW_spinbutton_new()
- Upper - the maximum number of the spinbutton
- Lower - the minimum number of the spinbutton

RESULT

No return result.

SEE ALSO

DW_spinbutton_set_pos, DW_spinbutton_get_pos

SOURCE

```
spinbutton = dw_spinbutton_new( 'Select Number', 0 )
Call dw_spinbutton_set_limits spinbutton, 200, 40
...
Call dw_spinbutton_set_pos spinbutton, 150
...
Say 'The spinbutton is at' dw_spinbutton_get_pos( spinbutton )
```

6.1.16.3. Spinbutton/DW_spinbutton_set_pos [Functions]

[Top] [Spinbutton] [Functions]

NAME

DW_spinbutton_set_pos

SYNOPSIS

dw_spinbutton_set_pos(Win, Position)

FUNCTION

Sets the position of the spinbutton.

ARGUMENTS

- Win - the window identifier returned from DW_spinbutton_new()
- Position - the position within the spinbutton's range

RESULT

No return result.

SEE ALSO

DW_spinbutton_get_pos

SOURCE

```
spinbutton = dw_spinbutton_new( 'Select Number', 0 )
Call dw_spinbutton_set_limits spinbutton, 200, 40
...
Call dw_spinbutton_set_pos spinbutton, 50
...
Say 'The spinbutton is at' dw_spinbutton_get_pos( spinbutton )
```

6.1.16.4. Spinbutton/DW_spinbutton_get_pos [Functions]

[Top] [Spinbutton] [Functions]

NAME

DW_spinbutton_get_pos

SYNOPSIS

pos = dw_spinbutton_get_pos(Win)

FUNCTION

Returns the current position of the spinbutton.

ARGUMENTS

- Win - the window identifier returned from DW_spinbutton_new()

RESULT

The position of the spinbutton.

SEE ALSO

DW_spinbutton_set_pos

SOURCE

```
spinbutton = dw_spinbutton_new( 'Select Number', 0 )
Call dw_spinbutton_set_limits spinbutton, 200, 40
...
Call dw_spinbutton_set_pos spinbutton, 150
...
Say 'The spinbutton is at' dw_spinbutton_get_pos( spinbutton )
```

6.1.17. Widgets/Splitbar [Modules]

[Top] [Widgets] [Modules]

DESCRIPTION

A **Splitbar** is another frame in which other widgets are packed. The **Splitbar** consists of two halves and a divider, which can be dragged by the user to resize each side of the splitbar. The widgets within each side of the **Splitbar** are automatically resized.

6.1.17.1. Splitbar/DW_splitbar_new [Functions]

[Top] [Splitbar] [Functions]

NAME

DW_splitbar_new

SYNOPSIS

win = dw_splitbar_new(Orientation, FirstBox, SecondBox, Id)

FUNCTION

Creates a new splitbar. A splitbar provides a widget for packing two boxes which can be dynamically resized by the user. This function also packs the boxes.

ARGUMENTS

- Orientation- !REXXDW.!DW_VERT to indicate a vertical box !REXXDW.!DW_HORZ to indicate a horizontal box
- FirstBox - the box to pack into the left or top half of the splitbar
- SecondBox - the box to pack into the right or bottom half of the splitbar
- Id - a numeric identifier used to identify the window

RESULT

A splitbar identifier.

SEE ALSO

WidgetOrientation

SOURCE

```
splitbar = dw_splitbar_new( !REXXDW.!DW_HORZ, box1, box2, 0 )
Call dw_splitbar_set splitbar, 40.0
```

6.1.17.2. Splitbar/DW_splitbar_set [Functions]

[Top] [Splitbar] [Functions]

NAME

DW_splitbar_set

SYNOPSIS

dw_splitbar_set(Win, Percent)

FUNCTION

Sets the relative sizes of the halves of the splitbar, by setting the percentage of the left or top half.

ARGUMENTS

- Win - the window identifier returned from DW_splitbar_new()
- Percent - the relative size of the left or top half of the splitbar

RESULT

No return result.

SOURCE

```
splitbar = dw_splitbar_new( !REXXDW.!DW_HORZ, box1, box2, 0 )
Call dw_splitbar_set splitbar, 40.0
```

6.1.17.3. Splitbar/DW_splitbar_get [Functions]

[Top] [Splitbar] [Functions]

NAME

DW_splitbat_get

SYNOPSIS

perc = dw_splitbar_get(Win)

FUNCTION

Returns the current percentage that the first half of the splitbar occupies.

ARGUMENTS

- Win - the window identifier returned from DW_splitbar_new()

RESULT

The percentage of the first half of the splitbar.

SEE ALSO

DW_splitbar_new, DW_splitbar_set

SOURCE

```
splitbar = dw_splitbar_new( !REXXDW.!DW_HORZ, box1, box2, 0 )
Call dw_splitbar_set splitbar, 40.0
...
Say 'The splitbar percentage is' dw_splitbar_get( splitbar )
```

6.1.18. Widgets/StaticText [Modules]

[Top] [Widgets] [Modules]

DESCRIPTION

StaticText widgets simply display static text.

6.1.18.1. StaticText/DW_text_new [Functions]

[Top] [StaticText] [Functions]

NAME

DW_text_new

SYNOPSIS

win = dw_text_new(Text, Id)

FUNCTION

Creates a new text window. A text window is a readonly window containing text.

ARGUMENTS

- Text - the text to appear in the window
- Id - a numeric identifier used to identify the window

RESULT

A text window identifier.

SEE ALSO

DW_status_text_new

SOURCE

```
text = dw_text_new( 'This is Rexx/DW', 0 )
```

6.1.18.2. StaticText/DW_status_text_new [Functions]

[Top] [StaticText] [Functions]

NAME

DW_status_text_new

SYNOPSIS

win = dw_status_text_new(Text, Id)

FUNCTION

Creates a new status text window. A status text window is a readonly window containing text with a sunken appearence.

ARGUMENTS

- Text - the text to appear in the window
- Id - a numeric identifier used to identify the window

RESULT

A status text window identifier.

SEE ALSO

DW_text_new

SOURCE

```
stext = dw_status_text_new( 'The current status is' status, 0 )
```

6.1.19. Widgets/Tree [Modules]

[Top] [Widgets] [Modules]

DESCRIPTION

A **Tree** widget provides a hierachical display of items. Paricularly suited to a filesystem display.

6.1.19.1. Tree/DW_tree_new [Functions]

[Top] [Tree] [Functions]

NAME

DW_tree_new

SYNOPSIS

win = dw_tree_new(Id)

FUNCTION

Creates a new tree widget. A tree widget allows the user to display a hierarchy in a tree format.

ARGUMENTS

- Id - a numeric identifier used to identify the window

RESULT

A tree identifier.

SOURCE

```
tree = dw_tree_new( 0 )
Do i = 1 To text.0
   item.i = dw_tree_insert_after( tree, topitem, Word( text.i ), ,
      fileicon, parent, 'item'i )
End
```

6.1.19.2. Tree/DW_tree_insert [Functions]

[Top] [Tree] [Functions]

NAME

DW_tree_insert

SYNOPSIS

data = dw_tree_insert(Win, Title[, Icon[, ParentItem[, ItemData]]])

FUNCTION

Inserts the tree item into the Tree at the end of any other items in the Tree with the same ParentItem. Intended to be used when populating and empty Tree

ARGUMENTS

- Win - the window identifier returned from DW_tree_new()
- Title - the text to appear next to the item
- Icon - the icon identifier returned from DW_icon_load_from_file() or DW_icon_load_from_data() to appear with the item
- ParentItem - the item identifier returned from **DW_tree_insert**() under which this item is to appear
- ItemData - The user data that is passed to the callback as ItemData argument

RESULT

A tree item identifier.

SEE ALSO

DW_tree_new, DW_tree_insert_after

SOURCE

```
tree = dw_tree_new( 0 )
Do i = 1 To text.0
   item.i = dw_tree_insert( tree, Word( text.i ), fileicon, parent, ,
      'item'i )
End
```

6.1.19.3. Tree/DW_tree_insert_after [Functions]

[Top] [Tree] [Functions]

NAME

DW_tree_insert_after

SYNOPSIS

data = dw_tree_insert_after(Win, Item, Title[, Icon[, ParentItem[, ItemData]]])

FUNCTION

Inserts the tree item into an existing Tree physically after the item specified in Item and at the level identified by ParentItem

ARGUMENTS

- Win - the window identifier returned from DW_tree_new()
- Item - the item in the tree where this item is to follow
- Title - the text to appear next to the item
- Icon - the icon identifier returned from DW_icon_load_from_file() or DW_icon_load_from_data() to appear with the item
- ParentItem - the item identifier returned from a previous call to **DW_tree_insert_after**() or DW_tree_insert() If not specified, the item will be added at the top level.
- ItemData - The user data that is passed to the callback as ItemData argument

RESULT

A tree item identifier.

SEE ALSO

DW_tree_new, DW_tree_insert

SOURCE

```
tree = dw_tree_new( 0 )
topitem = dw_tree_insert( tree, 'Parent' )
Do i = 1 To text.0
   item.i = dw_tree_insert( tree, topitem, Word( text.i ), fileicon, ,
      parent, 'item'i )
End
...
```

```
Call dw_tree_insert_after( tree, item.5, "New Item", fileicon, item.3 )
```

6.1.19.4. Tree/DW_tree_clear [Functions]

[Top] [Tree] [Functions]

NAME

DW_tree_clear

SYNOPSIS

dw_tree_clear(Win)

FUNCTION

Removes all items from a tree widget.

ARGUMENTS

- Win - the window identifier returned from DW_tree_new()

RESULT

No return result.

SEE ALSO

DW_tree_new, DW_tree_item_delete

SOURCE

```
tree = dw_tree_new( 0 )
...
Call dw_tree_clear tree
```

6.1.19.5. Tree/DW_tree_item_delete [Functions]

[Top] [Tree] [Functions]

NAME

DW_tree_item_delete

SYNOPSIS

dw_tree_item_delete(Win, Item)

FUNCTION

Deletes the specified Item in the tree. If the item is a parent item then all child items under it are also deleted. (Is this true???)

ARGUMENTS

- Win - the window identifier returned from DW_tree_new()
- Item - the item identifier returned from DW_tree_insert() or DW_tree_insert_after()

RESULT

No return result.

SEE ALSO

DW_tree_new, DW_tree_insert, DW_tree_insert_after

SOURCE

```
tree = dw_tree_new( 0 )
Do i = 1 To text.0
   item.i = dw_tree_insert( tree, Word( text.i ), fileicon )
End
...
Call dw_tree_item_delete tree, item.5
```

6.1.19.6. Tree/DW_tree_item_collapse [Functions]

[Top] [Tree] [Functions]

NAME

DW_tree_item_collapse

SYNOPSIS

dw_tree_item_collapse(Win, Item)

FUNCTION

Collapses the specified Item in the tree.

ARGUMENTS

- Win - the window identifier returned from DW_tree_new()
- Item - the item identifier returned from DW_tree_insert() or DW_tree_insert_after()

RESULT

No return result.

SEE ALSO

DW_tree_new, DW_tree_insert, DW_tree_insert_after, DW_tree_item_expand

SOURCE

```
tree = dw_tree_new( 0 )
Do i = 1 To text.0
   item.i = dw_tree_insert( tree, Word( text.i ), fileicon )
End
...
Call dw_tree_item_collapse tree, item.5
```

6.1.19.7. Tree/DW_tree_item_expand [Functions]

[Top] [Tree] [Functions]

NAME

DW_tree_item_expand

SYNOPSIS

dw_tree_item_expand(Win, Item)

FUNCTION

Expands the specified Item in the tree.

ARGUMENTS

- Win - the window identifier returned from DW_tree_new()
- Item - the item identifier returned from DW_tree_insert() or DW_tree_insert_after()

RESULT

No return result.

SEE ALSO

DW_tree_new, DW_tree_insert, DW_tree_insert_after, DW_tree_item_collapse

SOURCE

```
tree = dw_tree_new( 0 )
Do i = 1 To text.0
   item.i = dw_tree_insert( tree, Word( text.i ), fileicon )
End
...
```

```
Call dw_tree_item_expand tree, item.5
```

6.1.19.8. Tree/DW_tree_item_set_data [Functions]

[Top] [Tree] [Functions]

NAME

DW_tree_item_set_data

SYNOPSIS

dw_tree_item_set_data(Win, Item, UserData)

FUNCTION

Sets the userdata for the specified tree Item.

ARGUMENTS

- Win - the window identifier returned from DW_tree_new()
- Item - the item in the tree to which the data is to be attached
- UserData - the string to associate with the tree item's user data

RESULT

No return result.

SEE ALSO

DW_tree_new, DW_tree_item_get_data

SOURCE

```
tree = dw_tree_new( 0 )
Do i = 1 To text.0
   item.i = dw_tree_insert( tree, Word( text.i ), fileicon )
   Call dw_tree_item_set_data tree, item.i, 'item'i
End
...
Say 'The data from this item is' dw_tree_item_get_data( tree, item.5 ) ,
   'and should be "item5"'
```

6.1.19.9. Tree/DW_tree_item_get_data [Functions]

[Top] [Tree] [Functions]

NAME

DW_tree_item_get_data

SYNOPSIS

data = dw_tree_item_get_data(Win, Item)

FUNCTION

Returns the data the user set for this Item when calling DW_tree_item_set_data().

ARGUMENTS

- Win - the window identifier returned from DW_tree_new()
- Item - the item identifier returned from DW_tree_insert()

RESULT

The user supplied data.

SEE ALSO

DW_tree_new, DW_tree_item_set_data

SOURCE

```
tree = dw_tree_new( 0 )
Do i = 1 To text.0
   item.i = dw_tree_insert( tree, Word( text.i ), fileicon )
   Call dw_tree_item_set_data tree, item.i, 'item'i
End
...
Say 'The data from this item is' dw_tree_item_get_data( tree, item.5 ) ,
   'and should be "item5"'
```

6.1.19.10. Tree/DW_tree_item_select [Functions]

[Top] [Tree] [Functions]

NAME

DW_tree_item_select

SYNOPSIS

dw_tree_item_select(Win, Item)

FUNCTION

Makes the specified tree Item the currently selected item.

ARGUMENTS

- Win - the window identifier returned from DW_tree_new()
- Item - the item identifier returned from DW_tree_insert() or DW_tree_insert_after()

RESULT

No return result.

SEE ALSO

DW_tree_new, DW_tree_item_set_data, DW_tree_insert, DW_tree_insert_after

SOURCE

```
tree = dw_tree_new( 0 )
Do i = 1 To text.0
   item.i = dw_tree_insert( tree, Word( text.i ), fileicon )
   Call dw_tree_item_set_data tree, item.i, 'item'i
End
...
Call dw_tree_item_select( tree, item.5 )
```

6.1.19.11. Tree/DW_tree_item_change [Functions]

[Top] [Tree] [Functions]

NAME

DW_tree_item_change

SYNOPSIS

dw_tree_item_change(Win, Item, Title, Icon)

FUNCTION

Changes the Title and Icon for the specified tree Item.

ARGUMENTS

- Win - the window identifier returned from DW_tree_new()
- Item - the item in the tree where this item is to follow
- Title - the text to appear next to the item
- Icon - the icon identifier returned from DW_icon_load_from_file() or DW_icon_load_from_data() to appear with the item

RESULT

No return result.

SEE ALSO

DW_tree_new, DW_tree_insert, DW_tree_insert_after

SOURCE

```
tree = dw_tree_new( 0 )
Do i = 1 To text.0
   item.i = dw_tree_insert( tree, Word( text.i ), fileicon )
   Call dw_tree_item_set_data tree, item.i, 'item'i
End
...
Call dw_tree_item_change tree, item.5, 'New Title', diricon
```

6.1.19.12. Tree/DW_tree_get_parent [Functions]

[Top] [Tree] [Functions]

NAME

DW_tree_get_parent

SYNOPSIS

data = dw_tree_get_parent(Win, Item)

FUNCTION

Returns the parent of Item

ARGUMENTS

- Win - the window identifier returned from DW_tree_new()
- Item - the item identifier returned from DW_tree_insert()

RESULT

The parent of the supplied Item.

SEE ALSO

DW_tree_new, DW_tree_item_set_data

SOURCE

```
tree = dw_tree_new( 0 )
Do i = 1 To text.0
   item.i = dw_tree_insert( tree, Word( text.i ), fileicon )
   Call dw_tree_item_set_data tree, item.i, 'item'i
End
...
```

```
Say 'The parent of item.5 is' dw_tree_get_parent( tree, item.5 ) ,
   'and should be' item.4
```

6.1.20. Widgets/Window [Modules]

[Top] [Widgets] [Modules]

DESCRIPTION

A **Window** is frame in which other widgets are contained. It is also a generic term for a number of other widgets that can be manipulated by a number of functions in this section.

6.1.20.1. Window/DW_window_new [Functions]

[Top] [Window] [Functions]

NAME

DW_window_new

SYNOPSIS

win = dw_window_new(Win, Title, Flags)

FUNCTION

Creates a new window. A window is a visible frame containing a border and titlebar. It can also contain minimise/maximise/close buttons etc depending on the value of Flags.

ARGUMENTS

- Win - the window identifier returned from various functions that return a window identifier or DW_DESKTOP. This is this window's parent window
- Title - the text to appear in the titlebar of the window (if it has one)
- Flags - the flags defining the style and initial behaviour of the window

RESULT

A window identifier.

SEE ALSO

WindowStyleFlags

SOURCE

```
win = dw_window_new( !REXXDW.!DW_DESKTOP, 'Edit User Preferences', ,
```

```
windowstyle )
```

6.1.20.2. Window/DW_mdi_new [Functions]

[Top] [Window] [Functions]

NAME

DW_mdi_new

SYNOPSIS

win = dw_mdi_new(Id)

FUNCTION

Creates a new Multi-Document Interface (MDI) frame. An MDI frame is used as a container for multiple windows.

ARGUMENTS

- Id - a numeric identifier used to identify the window

RESULT

An MDI frame identifier.

SEE ALSO

DW_window_new

SOURCE

```
...
mdi = dw_mdi_new( 100 )
...
mdiwin1 = dw_window_new( mdi, 'First MDI window', style )
mdiwin2 = dw_window_new( mdi, 'Second MDI window', style )
...
```

6.1.20.3. Window/DW_window_destroy [Functions]

[Top] [Window] [Functions]

NAME

DW_window_destroy

SYNOPSIS

dw_window_destroy(Win)

FUNCTION

Destroys a window and all other items created within it.

ARGUMENTS

- Win - the window identifier returned from various functions that return a window identifier

RESULT

0 if successful, any other value is an error

SEE ALSO

DW_window_new

SOURCE

```
win = dw_window_new( !REXXDW.!DW_DESKTOP, 'Window on desktop', style )
...
Call dw_window_destroy( win )
```

6.1.20.4. Window/DW_window_capture [Functions]

[Top] [Window] [Functions]

NAME

DW_window_capture

SYNOPSIS

dw_window_capture(Win)

FUNCTION

Captures the mouse pointer to the specified window.

ARGUMENTS

- Win - the window identifier returned from various functions that return a window identifier

RESULT

No return result.

SEE ALSO

DW_window_release

NOTES

Only 1 window can capture the mouse pointer at any point in time.

SOURCE

```
Call dw_window_capture( win )
```

6.1.20.5. Window/DW_window_click_default [Functions]

[Top] [Window] [Functions]

NAME

DW_window_click_default

SYNOPSIS

dw_window_click_default(Win, DefaultWin)

FUNCTION

Defines the window; DefaultWin that is to receive a button click when the ENTER key is pressed in Win.

ARGUMENTS

- Win - the window identifier returned from a call to dw_window_new() used as the source of the ENTER key
- DefaultWin - the window identifier returned from various functions that return a window identifier which receives button click

RESULT

No return result.

NOTES

The source window; Win, should be a top-level window. Only one widget within a top-level window can be the widget to receive the click event. The DefaultWin should be a widget that accepts the DW_CLICKED_EVENT signal.

SOURCE

```
b1 = dw_button_new( 'OK', 0 )
...
Call dw_signal_connect b1, !REXXDW.!DW_CLICKED_EVENT, 'QuitCallback', !global.!mainwindow
```

```
...
Call dw_window_click_default( win, b1 )
```

6.1.20.6. Window/DW_window_default [Functions]

[Top] [Window] [Functions]

NAME

DW_window_default

SYNOPSIS

dw_window_default(Win, DefaultWin)

FUNCTION

Sets the default focus item; DefaultWin for the specified window.

ARGUMENTS

- Win - the window identifier returned from various functions that return a window identifier containing DefaultWin
- DefaultWin - the window identifier returned from various functions that return a window identifier which receives default focus

RESULT

No return result.

SOURCE

```
Call dw_window_default( win, mywin )
```

6.1.20.7. Window/DW_window_enable [Functions]

[Top] [Window] [Functions]

NAME

DW_window_enable

SYNOPSIS

dw_window_enable(Win)

FUNCTION

Enables a window.

ARGUMENTS

- Win - the window identifier returned from various functions that return a window identifier

RESULT

No return result.

SEE ALSO

DW_window_disable

SOURCE

```
win = dw_window_new( !REXXDW.!DW_DESKTOP, 'Window on desktop', style )
...
Call dw_window_disable( win )
...
Call dw_window_enable( win )
```

6.1.20.8. Window/DW_window_disable [Functions]

[Top] [Window] [Functions]

NAME

DW_window_disable

SYNOPSIS

dw_window_disable(Win)

FUNCTION

Disables a window.

ARGUMENTS

- Win - the window identifier returned from various functions that return a window identifier

RESULT

No return result.

SEE ALSO

DW_window_enable

SOURCE

```
win = dw_window_new( !REXXDW.!DW_DESKTOP, 'Window on desktop', style )
...
Call dw_window_disable( win )
```

6.1.20.9. Window/DW_window_from_id [Functions]

[Top] [Window] [Functions]

NAME

DW_window_from_id

SYNOPSIS

win = dw_window_from_id(Win, Id)

FUNCTION

Returns the window identifier from a widget within Win with the given Id.

ARGUMENTS

- Win - the window identifier returned from various functions that return a window identifier
- Id - the window id provided by the user in various function calls

RESULT

The window identifier.

NOTES

For this function to work, all Ids for widgets packed into the one window must be unique.

SOURCE

```
box = dw_box_new( !REXXDW.!DW_VERT, 0 )
bitmap = dw_bitmap_new( 100 )
Call dw_box_pack_start box, bitmap, 10, 10, !REXXDW.!DW_DONT_EXPAND_HORZ, ,
   !REXXDW.!DW_DONT_EXPAND_VERT, 0
button = dw_button_new( "Quit", 10 )
Call dw_box_pack_start box, button, 10, 10, !REXXDW.!DW_DONT_EXPAND_HORZ, ,
   !REXXDW.!DW_DONT_EXPAND_VERT, 0
...
Say 'Id' dw_window_from_id( box, 100 ) 'should be the same as' bitmap
```

6.1.20.10. Window/DW_window_show [Functions]

[Top] [Window] [Functions]

NAME

DW_window_show

SYNOPSIS

rc = dw_window_show(Win)

FUNCTION

Makes the specified window visible and sets focus to that window.

ARGUMENTS

- Win - the window identifier returned from various functions that return a window identifier

RESULT

0 if function succeeds, error for any other value

SEE ALSO

DW_window_hide

SOURCE

```
win = dw_window_new( !REXXDW.!DW_DESKTOP, 'Edit User Preferences', ,
   windowstyle )
...
Call dw_window_show( win )
```

6.1.20.11. Window/DW_window_hide [Functions]

[Top] [Window] [Functions]

NAME

DW_window_hide

SYNOPSIS

dw_window_hide(Win)

FUNCTION

Makes the specified window invisible.

ARGUMENTS

- Win - the window identifier returned from various functions that return a window identifier

RESULT

No return result.

SEE ALSO

DW_window_show

SOURCE

```
win = dw_window_new( !REXXDW.!DW_DESKTOP, 'Edit User Preferences', ,
   windowstyle )
...
Call dw_window_hide( win )
```

6.1.20.12. Window/DW_window_lower [Functions]

[Top] [Window] [Functions]

NAME

DW_window_lower

SYNOPSIS

dw_window_lower(Win)

FUNCTION

Makes the specified window appear behind all others.

ARGUMENTS

- Win - the window identifier returned from various functions that return a window identifier

RESULT

No return result.

SEE ALSO

DW_window_raise

SOURCE

```
win = dw_window_new( !REXXDW.!DW_DESKTOP, 'Edit User Preferences', ,
   windowstyle )
...
Call dw_window_lower( win )
```

6.1.20.13. Window/DW_window_minimize [Functions]

[Top] [Window] [Functions]

NAME

DW_window_minimize

SYNOPSIS

dw_window_minimize(Win)

FUNCTION

Minimises the specified window.

ARGUMENTS

- Win - the window identifier returned from various functions that return a window identifier

RESULT

No return result.

SEE ALSO

DW_window_maximize; but there isn't one!!

SOURCE

```
win = dw_window_new( !REXXDW.!DW_DESKTOP, 'Edit User Preferences', ,
   windowstyle )
...
Call dw_window_minimize( win )
```

6.1.20.14. Window/DW_window_raise [Functions]

[Top] [Window] [Functions]

NAME

DW_window_raise

SYNOPSIS

dw_window_raise(Win)

FUNCTION

Makes the specified window appear in front of all others.

ARGUMENTS

- Win - the window identifier returned from various functions that return a window identifier

RESULT

No return result.

SEE ALSO

DW_window_lower

SOURCE

```
win = dw_window_new( !REXXDW.!DW_DESKTOP, 'Edit User Preferences', ,
   windowstyle )
...
Call dw_window_raise( win )
```

6.1.20.15. Window/DW_window_redraw [Functions]

[Top] [Window] [Functions]

NAME

DW_window_redraw

SYNOPSIS

dw_window_redraw(Win)

FUNCTION

Redraws the specified window and all of widgets contained within it. This should be called after re-packing any widgets within the window.

ARGUMENTS

- Win - the window identifier returned from various functions that return a window identifier

RESULT

No return result.

SEE ALSO

DW_window_show

SOURCE

```
win = dw_window_new( !REXXDW.!DW_DESKTOP, 'Edit User Preferences', ,
   windowstyle )
...
Call dw_window_redraw( win )
```

6.1.20.16. Window/DW_window_release [Functions]

[Top] [Window] [Functions]

NAME

DW_window_release

SYNOPSIS

dw_window_release( )

FUNCTION

Releases the captured mouse pointer from a previous call to DW_window_captture().

ARGUMENTS

None

RESULT

No return result.

SEE ALSO

DW_window_capture

NOTES

Only 1 window can capture the mouse pointer at any point in time.

SOURCE

```
Call dw_window_capture( win )
```

```
...
Call dw_window_release( )
```

6.1.20.17. Window/DW_window_reparent [Functions]

[Top] [Window] [Functions]

NAME

DW_window_reparent

SYNOPSIS

dw_window_reparent(Win, ParentWin)

FUNCTION

Changes the parent window; ParentWin of the specified Win.

ARGUMENTS

- Win - the window identifier returned from various functions that return a window identifier to be reparented
- ParentWin - the window identifier returned from various functions that return a window to becoem the new parent. May also be DW_DESKTOP

RESULT

No return result.

SOURCE

```
mdi = dw_mdi_new( 100 )
win = dw_window_new( mdi, 'Edit User Preferences', windowstyle )
...
Call dw_window_reparent( win, !REXXDW.!DW_DESKTOP )
...
box1 = dw_box_new( !REXXDW.!DW_VERT, 0 )
Call dw_box_pack_start win, box1, 10, 10, !REXXDW.!DW_DONT_EXPAND_HORZ, ,
   !REXXDW.!DW_DONT_EXPAND_VERT, 0
button = dw_button_new( "Quit", 10 )
Call dw_box_pack_start box1, button, 10, 10, !REXXDW.!DW_DONT_EXPAND_HORZ, ,
   !REXXDW.!DW_DONT_EXPAND_VERT, 0
...
box2 = dw_box_new( !REXXDW.!DW_VERT, 0 )
Call dw_window_reparent box2, button
```

6.1.20.18. Window/DW_window_set_bitmap [Functions]

[Top] [Window] [Functions]

NAME

DW_window_set_bitmap

SYNOPSIS

dw_window_set_bitmap(Win, Id, Filename)

FUNCTION

Associates the bitmap in Filename with the specified window.

ARGUMENTS

- Win - the window identifier returned from various functions that return a window identifier
- Id - a numeric identifier used to identify the window
- Filename - the name of a file containing a valid bitmap image (.BMP, or .XPM)

RESULT

No return result.

SEE ALSO

DW_bitmap_new_from_file, DW_bitmap_new_from_data

SOURCE

```
...
bitmap = dw_bitmap_new( 100 )
Call dw_box_pack_start win, bitmap, 10, 10, !REXXDW.!DW_DONT_EXPAND_HORZ, ,
   !REXXDW.!DW_DONT_EXPAND_VERT, 0
Call dw_window_set_bitmap bitmap, 0, 'mypicture'
```

6.1.20.19. Window/DW_window_set_bitmap_from_data [Functions]

[Top] [Window] [Functions]

NAME

DW_window_set_bitmap_from_data

SYNOPSIS

dw_window_set_bitmap_from_data(Win, Id, Data)

FUNCTION

Associates the bitmap in Data with the specified window.

ARGUMENTS

- Win - the window identifier returned from various functions that return a window identifier
- Id - a numeric identifier used to identify the window
- Data - the contents of the image.

RESULT

No return result.

SOURCE

```
...
ico = "0000A6D7007E"
...
bitmap = dw_bitmap_new( 100 )
Call dw_box_pack_start win, bitmap, 10, 10, !REXXDW.!DW_DONT_EXPAND_HORZ, ,
   !REXXDW.!DW_DONT_EXPAND_VERT, 0
Call dw_window_set_bitmap_from_data bitmap, 0, x2c( ico )
```

6.1.20.20. Window/DW_window_set_border [Functions]

[Top] [Window] [Functions]

NAME

DW_window_set_border

SYNOPSIS

dw_window_set_border(Win, Width)

FUNCTION

Sets the width of the border for the specified window.

ARGUMENTS

- Win - the window identifier returned from various functions that return a window identifier
- Width - the width in pixels of the window's border

RESULT

0 if successful, any other value is an error

NOTES

This function only works on OS/2.

SOURCE

```
win = dw_window_new( mdi, 'Edit User Preferences', windowstyle )
Call dw_window_set_border win, 20
```

6.1.20.21. Window/DW_window_set_pointer [Functions]

[Top] [Window] [Functions]

NAME

DW_window_set_pointer

SYNOPSIS

dw_window_set_pointer(Win, Pointer)

FUNCTION

Changes the mouse pointer to Pointer.

ARGUMENTS

- Win - the window identifier returned from various functions that return a window identifier
- Pointer - the new pointer; one of PointerTypes

RESULT

No return result.

SEE ALSO

PointerTypes

NOTES

To ensure that the pointer is changed immediately, call DW_main_sleep() afterwards for a short period, like 10 milliseconds

SOURCE

```
win = dw_window_new( !REXXDW.!DW_DESKTOP, 'Edit User Preferences', ,
   windowstyle )
...
Call dw_window_set_pointer( win, !REXXDW.!DW_POINTER_CLOCK )
Call dw_main_sleep 10
```

6.1.20.22. Window/DW_window_set_color [Functions]

[Top] [Window] [Functions]

NAME

DW_window_set_color

SYNOPSIS

dw_window_set_color(Win, Fore, Back)

FUNCTION

Sets the foreground and background colour of the window.

ARGUMENTS

- Win - the window identifier returned from various functions that return a window identifier
- Fore - the internal RexxDW colour for the foreground
- Back - the internal RexxDW colour for the background

RESULT

0 if successful, any other value is an error

SEE ALSO

Colours, DW_rgb

SOURCE

```
text = dw_text_new( 'File to browse', 0 )
Call dw_window_set_color text, !REXXDW.!DW_CLR_BLACK, !REXXDW.!DW_CLR_YELLOW
```

6.1.20.23. Window/DW_window_set_focus [Functions]

[Top] [Window] [Functions]

NAME

DW_window_set_focus

SYNOPSIS

dw_window_set_focus(Win)

FUNCTION

Makes the specified Win the active window.

ARGUMENTS

- Win - the window identifier returned from various functions that return a window identifier

RESULT

No return result.

SOURCE

```
entry = dw_entryfield_new( '', 0 )
...
Call dw_window_set_focus entry
```

6.1.20.24. Window/DW_window_set_font [Functions]

[Top] [Window] [Functions]

NAME

DW_window_set_font

SYNOPSIS

dw_window_set_font(Win, Fontname)

FUNCTION

Sets the text font of the window.

ARGUMENTS

- Win - the window identifier returned from various functions that return a window identifier
- Fontname - the font name of the desired font

RESULT

0 if successful, any other value is an error

SEE ALSO

Fontnames, DW_window_get_font

NOTES

Font names in RexxDW are platform dependent. See Fontnames for details on how to specify fonts on each platform.

SOURCE

```
Parse Source os . prog
Select
   When os = 'OS2' | os = 'OS/2' Then  FIXEDFONT = "5.System VIO"
   When os = 'WIN32' Then FIXEDFONT = "10.Terminal"
   When !REXXDW.!GTK_MAJOR_VERSION > 1 Then FIXEDFONT = "monospace 10"
   Otherwise FIXEDFONT = "fixed"
End
text = dw_text_new( 'File to browse', 0 )
Call dw_window_set_font text, FIXEDFONT
```

6.1.20.25. Window/DW_window_get_font [Functions]

[Top] [Window] [Functions]

NAME

DW_window_get_font

SYNOPSIS

font = dw_window_get_font(Win)

FUNCTION

Returns the font details from the specified window.

ARGUMENTS

- Win - the window identifier returned from various functions that return a window identifier

RESULT

The value of the font details of the window in Dynamic Windows font format: fontsize.fontname [bold] [italic]

SEE ALSO

Fontnames, DW_window_set_font

SOURCE

```
entryfield = dw_entryfield_new( "Initial value", 0 )
...
Say 'The font for entryfield is' dw_window_get_font( entryfield )
```

6.1.20.26. Window/DW_window_set_icon [Functions]

[Top] [Window] [Functions]

NAME

DW_window_set_icon

SYNOPSIS

dw_window_set_icon(Win, Icon)

FUNCTION

Sets the text font of the window.

ARGUMENTS

- Win - the window identifier returned from various functions that return a window identifier
- Icon - the icon identifier returned from DW_icon_load_from_file() or DW_icon_load_from_data()

RESULT

No return result.

SEE ALSO

DW_icon_load_from_file, DW_icon_load_from_data

SOURCE

```
rexxdwicon = dw_icon_load_from_file( '/home/mark/rexxdw' )
win = dw_window_new( mdi, 'Edit User Preferences', windowstyle )
Call dw_window_set_icon win, rexxdwicon
```

6.1.20.27. Window/DW_window_set_gravity [Functions]

[Top] [Window] [Functions]

NAME

DW_window_set_gravity

SYNOPSIS

dw_window_set_gravity(Win, Horz, Vert)

FUNCTION

Sets the horizontal and vertical positions from which dw_window_set_pos() values are relative to.

ARGUMENTS

- Win - the window identifier returned from DW_window_new()
- Horz - Any value specified in WindowGravityValues for x position
- Vert - Any value specified in WindowGravityValues for y position

RESULT

No return result.

SEE ALSO

WindowGravityValues, DW_window_set_pos, DW_window_set_pos_size

SOURCE

```
win = dw_window_new( !REXXDW.!DW_DESKTOP, 'Window on desktop', style )
Call dw_window_set_gravity win, !REXXDW.!DW_GRAV_LEFT, !REXXDW.!DW_GRAV_LEFT
Call dw_window_set_pos win, 0, 0 -- window positioned in top left corner of screen
...
```

6.1.20.28. Window/DW_window_set_pos [Functions]

[Top] [Window] [Functions]

NAME

DW_window_set_pos

SYNOPSIS

dw_window_set_pos(Win, X, Y)

FUNCTION

Sets the position of the top left corner of the window relative to the window's parent.

ARGUMENTS

- Win - the window identifier returned from various functions that return a window identifier
- X - the x coordinate in pixels of the top left corner
- Y - the y coordinate in pixels of the top left corner

RESULT

No return result.

SEE ALSO

DW_window_set_pos_size, DW_window_set_size

NOTES

Some window managers (at least under X11) will not respect a request for the position of a window.

SOURCE

```
win = dw_window_new( !REXXDW.!DW_DESKTOP, 'Edit User Preferences', ,
   windowstyle )
Call dw_window_set_pos win, 50, 90
```

6.1.20.29. Window/DW_window_set_size [Functions]

[Top] [Window] [Functions]

NAME

DW_window_set_size

SYNOPSIS

dw_window_set_size(Win, Width, Height)

FUNCTION

Sets the size of the window in pixels.

ARGUMENTS

- Win - the window identifier returned from various functions that return a window identifier
- Width - the requested width of Window in pixels
- Height - the requested height of Window in pixels

RESULT

No return result.

SEE ALSO

DW_window_set_pos, DW_window_set_pos_size

NOTES

Some window managers (at least under X11) will not respect a request for the size of a window.

SOURCE

```
win = dw_window_new( !REXXDW.!DW_DESKTOP, 'Edit User Preferences', ,
   windowstyle )
Call dw_window_set_size win, 640, 480
```

6.1.20.30. Window/DW_window_set_pos_size [Functions]

[Top] [Window] [Functions]

NAME

DW_window_set_pos_size

SYNOPSIS

dw_window_set_pos_size(Win, X, Y, Width, Height)

FUNCTION

Sets the position of the top left corner of the window relative to the window's parent and the size of the window in pixels.

ARGUMENTS

- Win - the window identifier returned from various functions that return a window identifier
- X - the x coordinate in pixels of the top left corner
- Y - the y coordinate in pixels of the top left corner
- Width - the requested width of Window in pixels
- Height - the requested height of Window in pixels

RESULT

No return result.

SEE ALSO

DW_window_set_pos, DW_window_set_size

NOTES

Some window managers (at least under X11) will not respect a request for the position of a window.

SOURCE

```
win = dw_window_new( !REXXDW.!DW_DESKTOP, 'Edit User Preferences', ,
   windowstyle )
Call dw_window_set_pos_size win, 50, 90, 640, 480
```

6.1.20.31. Window/DW_window_get_pos_size [Functions]

[Top] [Window] [Functions]

NAME

DW_window_get_pos_size

SYNOPSIS

- X Y Width Height = dw_window_get_pos_size(Win)

FUNCTION

Returns the X and Y coordinates of the specified window, and the Width and Height of the window.

ARGUMENTS

- Win - the window identifier returned from various functions that return a window identifier

RESULT

X, Y, Width and Height are returned as four words. PARSE VALUE is the easiest way to obtain all values.

SOURCE

```
...
win = dw_window_new( !REXXDW.!DW_DESKTOP, 'Edit User Preferences', ,
   windowstyle )
...
Parse Value dw_window_get_pos_size( win ) With x y width height
```

6.1.20.32. Window/DW_window_get_preferred_size [Functions]

[Top] [Window] [Functions]

NAME

DW_window_get_preferred_size

SYNOPSIS

- Width Height = dw_window_get_preferred_size(Win)

FUNCTION

Returns the Width and Height the system thinks the window should be.

ARGUMENTS

- Win - the window identifier returned from various functions that return a window identifier

RESULT

Width and Height are returned as two words. PARSE VALUE is the easiest way to obtain all values.

SOURCE

```
...
win = dw_window_new( !REXXDW.!DW_DESKTOP, 'Edit User Preferences', ,
   windowstyle )
...
Parse Value dw_window_get_preferred_size( win ) With width height
```

6.1.20.33. Window/DW_window_set_style [Functions]

[Top] [Window] [Functions]

NAME

DW_window_set_style

SYNOPSIS

dw_window_set_style(Win, Style, Mask)

FUNCTION

Changes the attributes of text within the specified window.

ARGUMENTS

- Win - the window identifier returned from various functions that return a window identifier
- Style - Any value specified in WindowAttributeFlags ORed together
- Mask - Any value specified in WindowAttributeFlags ORed together

RESULT

No return result.

SEE ALSO

WindowAttributeFlags

SOURCE

```
win = dw_text_new( 'Edit User Preferences' )
style = dw_or( !REXXDW.!DW_DT_RIGHT, !REXXDW.!DW_DT_VCENTER )
mask =  dw_or( !REXXDW.!DW_DT_RIGHT, !REXXDW.!DW_DT_VCENTER )
Call dw_window_set_style win, style, mask
```

6.1.20.34. Window/DW_window_set_tooltip [Functions]

[Top] [Window] [Functions]

NAME

DW_window_set_tooltip

SYNOPSIS

dw_window_set_tooltip(Win[, Text])

FUNCTION

Sets the floating help text for the specified window.

ARGUMENTS

- Win - the window identifier returned from various functions that return a window identifier
- Text - the value of the floating help text for the window or if omitted disables the floating help text

RESULT

No return result.

SOURCE

```
entryfield = dw_entryfield_new( "Initial value", 0 )
...
Call dw_window_set_tooltip entryfield, 'Rexx is king'
```

6.1.20.35. Window/DW_window_set_text [Functions]

[Top] [Window] [Functions]

NAME

DW_window_set_text

SYNOPSIS

dw_window_set_text(Win, Text)

FUNCTION

Sets the text for the specified window. If the window is a top level window and it has a title bar, the title bar text is changed.

ARGUMENTS

- Win - the window identifier returned from various functions that return a window identifier
- Text - the value of the text attribute the window is to have

RESULT

No return result.

SEE ALSO

DW_window_get_text

NOTES

Not all window types have a meaningful text attribute. Those that do are:

Text	DW_text_new
StatusText	DW_statustext_new
Combobox	DW_combobox_new()
EntryField	DW_entryfield_new()
EntryFieldPassword	DW_entryfield_password_new()

SOURCE

```
entryfield = dw_entryfield_new( "Initial value", 0 )
...
Call dw_window_set_text entryfield, 'Rexx is king'
```

6.1.20.36. Window/DW_window_get_text [Functions]

[Top] [Window] [Functions]

NAME

DW_window_get_text

SYNOPSIS

text = dw_window_get_text(Win)

FUNCTION

Returns the text attribute from the specified window.

ARGUMENTS

- Win - the window identifier returned from various functions that return a window identifier

RESULT

The value of the text attribute of the window.

SEE ALSO

DW_window_set_text

NOTES

Not all windows have a meaningful text attribute. Those that do are:

Text	DW_text_new
StatusText	DW_statustext_new
Combobox	DW_combobox_new()
EntryField	DW_entryfield_new()
EntryFieldPassword	DW_entryfield_password_new()

SOURCE

```
entryfield = dw_entryfield_new( "Initial value", 0 )
...
Say 'The text is now' dw_window_get_text( entryfield )
```

6.1.20.37. Window/DW_window_set_data [Functions]

[Top] [Window] [Functions]

NAME

DW_window_set_data

SYNOPSIS

dw_window_set_data(Win, Name, Value)

FUNCTION

Sets the user variable; Name to the specified Value for the specified window.

ARGUMENTS

- Win - the window identifier returned from various functions that return a window identifier
- Name - the user variable name
- Value - the value that Name is to be set to

RESULT

No return result.

SEE ALSO

DW_window_get_data

NOTES

Don't use variable names that start with "_dw"; they are used internally by Dynamic Windows.

SOURCE

```
entryfield = dw_entryfield_new( 'Initial value', 0 )
Call dw_window_set_data entryfield, 'MyVar', 'my useful data'
...
Say 'The value of "MyVar" is:' dw_window_get_data( entryfield, 'MyVar' )
```

6.1.20.38. Window/DW_window_get_data [Functions]

[Top] [Window] [Functions]

NAME

DW_window_get_data

SYNOPSIS

data = dw_window_get_data(Win, Name)

FUNCTION

Returns the user variable data for the window with the given Name created by a call to DW_window_set_data();

ARGUMENTS

- Win - the window identifier returned from various functions that return a window identifier
- Name - the user variable name

RESULT

The value of the user variable Name for the window

SEE ALSO

DW_window_set_data

NOTES

Don't use variable names that start with "_dw"; they are used internally by Dynamic Windows.

SOURCE

```
entryfield = dw_entryfield_new( 'Initial value', 0 )
```

```
Call dw_window_set_data entryfield, 'MyVar', 'my useful data'
...
Say 'The value of "MyVar" is:' dw_window_get_data( entryfield, 'MyVar' )
```

6.1.21. Widgets/Miscellaneous [Modules]

[Top] [Widgets] [Modules]

DESCRIPTION

There always has to be things that can't be logically grouped together. These are those.

6.1.21.1. Miscellaneous/DW_messagebox [Functions]

[Top] [Miscellaneous] [Functions]

NAME

DW_messagebox

SYNOPSIS

action = dw_messagebox(Title, Flags, Text)

FUNCTION

Displays a modal messagebox in the middle of the desktop.

ARGUMENTS

- Title - the window title for the messagebox
- Flags - indicates the buttons to be incorporated into the messagebox and the icon to be displayed
- Text - the text inside the messagebox

RESULT

Depending on the button clicked, one of the values defined in MessageboxResults

NOTES

The maximum length of Text is 1023 characters.

SEE ALSO

MessageboxFlags, MessageboxResults

SOURCE

```
...
action = dw_messagebox( 'Are you sure?', dw_or( !REXXDW.!DW_MB_YESNO, ,
   !REXXDW.!DW_MB_QUESTION ), 'Delete selected files' )
If action = !REXXDW.!DW_MB_RESULT_YES Then Call DeleteFiles
```

6.1.21.2. Miscellaneous/DW_bitmap_new [Functions]

[Top] [Miscellaneous] [Functions]

NAME

DW_bitmap_new

SYNOPSIS

win = dw_bitmap_new(Id)

FUNCTION

Creates a new bitmap window used as the container for a bitmap image.

ARGUMENTS

- Id - a numeric identifier used to identify the window

RESULT

A window identifier.

SEE ALSO

DW_window_set_bitmap, DW_bitmap_new_from_file, DW_bitmap_new_from_data

SOURCE

```
...
bitmap = dw_bitmap_new( 100 )
Call dw_box_pack_start win, bitmap, 10, 10, !REXXDW.!DW_DONT_EXPAND_HORZ, ,
   !REXXDW.!DW_DONT_EXPAND_VERT, 0
Call dw_window_set_bitmap bitmap, 0, 'mypicture'
```

6.1.21.3. Miscellaneous/DW_bitmap_new_from_file [Functions]

[Top] [Miscellaneous] [Functions]

NAME

DW_bitmap_new_from_file

SYNOPSIS

win = dw_bitmap_new_from_file(Id, Filename)

FUNCTION

Creates a new bitmap window including a bitmap image.

ARGUMENTS

- Id - a numeric identifier used to identify the window
- Filename - the filename of the bitmap image

RESULT

A window identifier.

SEE ALSO

DW_window_set_bitmap, DW_bitmap_new, DW_bitmap_new_from_data

SOURCE

```
...
bitmap = dw_bitmap_new_from_file( 100, 'mypicture' )
Call dw_box_pack_start win, bitmap, 10, 10, !REXXDW.!DW_DONT_EXPAND_HORZ, ,
   !REXXDW.!DW_DONT_EXPAND_VERT, 0
```

6.1.21.4. Miscellaneous/DW_bitmap_new_from_data [Functions]

[Top] [Miscellaneous] [Functions]

NAME

DW_bitmap_new_from_data

SYNOPSIS

win = dw_bitmap_new_from_data(Id, Data)

FUNCTION

Creates a new bitmap window including a bitmap image from a memory image.

ARGUMENTS

- Id - a numeric identifier used to identify the window
- Data - the contents of the image.

RESULT

A window identifier.

SEE ALSO

DW_bitmap_new_from_file

SOURCE

```
...
ico = "0000A6D7007E"
...
bitmap = dw_bitmap_new_from_data( 100, x2c( ico ) )
Call dw_box_pack_start win, bitmap, 10, 10, !REXXDW.!DW_DONT_EXPAND_HORZ, ,
   !REXXDW.!DW_DONT_EXPAND_VERT, 0
```

6.1.21.5. Miscellaneous/DW_icon_load_from_file [Functions]

[Top] [Miscellaneous] [Functions]

NAME

DW_icon_load_from_file

SYNOPSIS

win = dw_icon_load_from_file(Filename)

FUNCTION

Creates a new icon identifier for use in other functions from the file specified in Filename. Supported icon types are Windows icons (.ico), OS/2 icons(.ico) and pixmaps(.xpm) on GTK. GTK 2.x also supports Windows icons (.ico).

ARGUMENTS

- Filename - the filename of the icon file

RESULT

An icon identifier.

NOTES

The Filename can contain a path component. For portability, do not specify the file's extension; let dwindows work this out.

SOURCE

```
normalfileicon = dw_icon_load_from_file( '/home/mark/normalfile' )
```

6.1.21.6. Miscellaneous/DW_icon_load_from_data [Functions]

[Top] [Miscellaneous] [Functions]

NAME

DW_icon_load_from_data

SYNOPSIS

win = dw_icon_load_from_data(Data)

FUNCTION

Creates a new icon identifier for use in other functions from the file specified in Filename. Supported icon types are Windows icons (.ico), OS/2 icons(.ico) and pixmaps(.xpm) on GTK. GTK 2.x also supports Windows icons (.ico).

ARGUMENTS

- Data - the contents of the image.

RESULT

An icon identifier.

NOTES

The Data argument is to contain an exact image of the icon.

SOURCE

```
...
ico = "0000A6D7007E"
...
normalfileicon = dw_icon_load_from_data( xc2( ico ) )
```

6.1.21.7. Miscellaneous/DW_taskbar_insert [Functions]

[Top] [Miscellaneous] [Functions]

NAME

DW_taskbar_insert

SYNOPSIS

win = dw_taskbar_insert(Toplevel, Icon[, Text[, Callback[, UserData]]])

FUNCTION

Inserts an icon into the window manager's taskbar.

ARGUMENTS

- Toplevel - the window handle for the application's toplevel window
- Icon - the icon handle returned from calls like DW_icon_from_data to be displayed on the taskbar
- Text - (optional) the text to be displayed when the mouse hovers over the icon
- Function - (optional) the Rexx procedure that is called when the a mouse button is pressed on the taskbar icon
- UserData - (optional) user supplied data to pass to the Function

RESULT

Returns a window identifier for the taskbar icon

NOTES

If you want a context menu displayed when a button is clicked on the taskbar icon, you need to specify Function, that Function must conform to ButtonPressEventCallback, and the callback must display a popup menu

SEE ALSO

DW_taskbar_delete, ButtonPressEventCallback

SOURCE

```
...
ico = "0000A6D7007E"
...
icon = dw_icon_load_from_data( xc2( ico ) )
...
win = dw_taskbar_insert( toplevel, icon, ,'contextmenu_cb', 'fred', 'mary' )
...
contextmenu_cb:
Parse Arg win, x, y, button
Say 'Button:' button 'pressed at:' x'/'y 'in Window:' win
If button \= 1 Then Return 0
menu = dw_menu_new( 0 )
menuitem = dw_menu_append_item( menu, '~Show Window', 1011, 0, ,
   !REXXDW.!DW_MENU_END, !REXXDW.!DW_MENU_NOT_CHECKABLE, 0 )
...
Call dw_menu_popup menu, win, x, y
Return 1
```

6.1.21.8. Miscellaneous/DW_taskbar_delete [Functions]

[Top] [Miscellaneous] [Functions]

NAME

DW_taskbar_delete

SYNOPSIS

dw_taskbar_delete(Win, Icon)

FUNCTION

Deletes an icon into the window manager's taskbar.

ARGUMENTS

- Win - the window handle returned from a call to DW_taskbar_insert
- Icon - the icon handle used in the call to DW_taskbar_insert

RESULT

None

SEE ALSO

DW_taskbar_insert

SOURCE

```
...
win = dw_taskbar_insert( toplevel, icon )
...
Call dw_taskbar_delete win, icon
```

6.2. Functions/ProcessControl [Modules]

[Top] [Functions] [Modules]

DESCRIPTION

These functions control the execution of the current process or instigate other processes.

6.2.1. ProcessControl/DW_init [Functions]

[Top] [ProcessControl] [Functions]

NAME

DW_init

SYNOPSIS

rc = dw_init( )

FUNCTION

Initialises the Rexx/DW interface. This MUST be called before any other Rexx/DW function.

ARGUMENTS

None

RESULT

0 if all OK, non-zero if an error occurred

SOURCE

```
Call dw_init
```

6.2.2. ProcessControl/DW_main [Functions]

[Top] [ProcessControl] [Functions]

NAME

DW_main

SYNOPSIS

dw_main( )

FUNCTION

This is the main event dispatcher function for Rexx interpreters which support callbacks.

ARGUMENTS

None

RESULT

No return result.

NOTES

This function can only be called if your Rexx interpreter supports the RexxCallback() API. This can be determined by the value of the variable !REXXDW.!HAVE_REXXCALLBACK being 1.

SEE ALSO

DW_main_iteration

SOURCE

```
...
If !REXXDW.!HAVE_REXXCALLBACK Then Call dw_main()
Else
   Do Forever
      cmd = dw_main_iteration()
      If cmd \= '' Then Interpret 'Call' cmd
   End
Return 0
```

6.2.3. ProcessControl/DW_main_iteration [Functions]

[Top] [ProcessControl] [Functions]

NAME

DW_main_iteration

SYNOPSIS

cmd = dw_main_iteration( )

FUNCTION

This is the main event dispatcher function for Rexx interpreters which DO NOT support callbacks.

ARGUMENTS

None

RESULT

The Rexx function called back, follwoed by parameters

NOTES

This function is supplied for interpreters that do not support the RexxCallback() API. This can be determined by the value of the variable !REXXDW.!HAVE_REXXCALLBACK being 1. You have to be very careful with any optional data you pass to a callback function as the data could be "interpreted" with syntax errors.

SEE ALSO

DW_main

SOURCE

```
...
If !REXXDW.!HAVE_REXXCALLBACK Then Call dw_main()
Else
   Do Forever
      cmd = dw_main_iteration()
      If cmd \= '' Then Interpret 'Call' cmd
   End
Return 0
```

6.2.4. ProcessControl/DW_main_sleep [Functions]

[Top] [ProcessControl] [Functions]

NAME

DW_main_sleep

SYNOPSIS

dw_main_sleep(Period)

FUNCTION

This function dispatches any events in the event queue for the number of milliseconds specified by Period.

ARGUMENTS

- Period - the number of milliseconds the event dispatcher runs for

RESULT

No return result.

NOTES

This function is useful to ensure that any outstanding event is executed before continuing execution within the code and not having to wait until control passes back to DW_Main() or DW_main_iteration().

SEE ALSO

DW_main, DW_main_iteration

SOURCE

```
...
Call dw_window_set_pointer mainwindow, !REXXDW.!DW_POINTER_CLOCK
Call dw_main_sleep 10
```

6.2.5. ProcessControl/DW_main_quit [Functions]

[Top] [ProcessControl] [Functions]

NAME

DW_main_quit

SYNOPSIS

dw_main_quit( )

FUNCTION

This function terminates the call to DW_Main().

ARGUMENTS

None

RESULT

No return result.

NOTES

This function is useful to allow program control to continue after the call to DW_Main(). It would normally be called in a callback associated with a toplevel window close event.

SEE ALSO

DW_main, DW_main_iteration

SOURCE

```
...
Call dw_main
Say 'do cleanup tasks here'
Exit 0
...
QuitCallback: Procedure Expose !REXXDW.
Call dw_main_quit
Return 0
```

...

6.2.6. ProcessControl/DW_shutdown [Functions]

[Top] [ProcessControl] [Functions]

NAME

DW_shutdown

SYNOPSIS

dw_shutdown( )

FUNCTION

This function cleans up all dwindows resources in preparation for a Rexx Exit

ARGUMENTS

None

RESULT

No return result.

SEE ALSO

DW_exit

SOURCE

```
...
Call dw_main
Say 'do cleanup tasks here'
Call dw_shutdown
Exit 0
...
QuitCallback: Procedure Expose !REXXDW.
Call dw_main_quit
Return 0
...
```

6.2.7. ProcessControl/DW_exit [Functions]

[Top] [ProcessControl] [Functions]

NAME

DW_exit

SYNOPSIS

dw_exit(Rcode)

FUNCTION

This function cleans up all dwindows resources and exits the process with a return code of Rcode.

ARGUMENTS

Rcode - numeric return value

RESULT

No return result.

NOTES

This function is a call to DW_shutdown followed by EXIT.

SEE ALSO

DW_shutdown

SOURCE

```
...
Call dw_main
Say 'do cleanup tasks here'
Call dw_exit 0
Exit 0 -- not reached
...
QuitCallback: Procedure Expose !REXXDW.
Call dw_main_quit
Return 0
...
```

6.3. Functions/Dialog [Modules]

[Top] [Functions] [Modules]

DESCRIPTION

A **Dialog** allows the programmer to create an interface with the user that is controlled outside of the normal main message dispatcher. other processes.

6.3.1. Dialog/DW_dialog_new [Functions]

[Top] [Dialog] [Functions]

NAME

DW_dialog_new

SYNOPSIS

dialog = dw_dialog_new( )

FUNCTION

Creates a new dialog. A dialog is a control widget (as opposed to a display widget) that allows the programmer to create a new toplevel window and have the user interact with the contents of the window, independently of any other toplevel window.

ARGUMENTS

None

RESULT

Returns a dialog identifier

SEE ALSO

DW_dialog_dismiss, DW_dialog_wait

NOTES

Due to the lack of real callback capabilities with Open Object Rexx (ooRexx 3.x) any callbacks specified using dw_signal_connect() from a dialog window will not work. Only Regina 3.3 or above will operate correctly with dialogs.

SOURCE

```
...
dialog_wait = dw_dialog_new( )
Call dw_signal_connect win, !REXXDW.!DW_DELETE_EVENT, ,
   'GenericCloseCallback', win, dialog_wait
...
result = dw_dialog_wait( dialog_wait )
If result = 'cancel' Then Return 1
...
GenericCloseCallback: Procedure Expose !REXXDW.
Parse Arg ., window, dialog
Call dw_window_destroy window
If dialog \= '' Then Call dw_dialog_dismiss dialog, 'close'
Return 0
```

6.3.2. Dialog/DW_dialog_dismiss [Functions]

[Top] [Dialog] [Functions]

NAME

DW_dialog_dismiss

SYNOPSIS

dw_dialog_dismiss(Dialog, ReturnString)

FUNCTION

Prepares a dialog window for exiting and returns ReturnString to the waiting DW_dialog_wait().

ARGUMENTS

- Dialog - an identifier returned from a call to DW_dialog_new()
- ReturnString - the string to return to DW_dialog_wait()

RESULT

Always returns 0

SEE ALSO

DW_dialog_new, DW_dialog_wait

SOURCE

```
...
dialog_wait = dw_dialog_new( )
Call dw_signal_connect win, !REXXDW.!DW_DELETE_EVENT, ,
   'GenericCloseCallback', win, dialog_wait
...
result = dw_dialog_wait( dialog_wait )
If result = 'cancel' Then Return 1
...
GenericCloseCallback: Procedure Expose !REXXDW.
Parse Arg ., window, dialog
Call dw_window_destroy window
If dialog \= '' Then Call dw_dialog_dismiss dialog, 'close'
Return 0
```

6.3.3. Dialog/DW_dialog_wait [Functions]

[Top] [Dialog] [Functions]

NAME

DW_dialog_wait

SYNOPSIS

returnstring = dw_dialog_wait(Dialog)

FUNCTION

Creates a new dialog for handling modal dialogs.

ARGUMENTS

- Dialog - an identifier returned from a call to DW_dialog_new()

RESULT

Returns the string set by the call to DW_dialog_dismiss()

SEE ALSO

DW_dialog_dismiss, DW_dialog_new

SOURCE

```
...
dialog_wait = dw_dialog_new( )
Call dw_signal_connect win, !REXXDW.!DW_DELETE_EVENT, ,
   'GenericCloseCallback', win, dialog_wait
...
result = dw_dialog_wait( dialog_wait )
If result = 'cancel' Then Return 1
...
GenericCloseCallback: Procedure Expose !REXXDW.
Parse Arg ., window, dialog
Call dw_window_destroy window
If dialog \= '' Then Call dw_dialog_dismiss dialog, 'close'
Return 0
```

6.4. Functions/CallbackManagement [Modules]

[Top] [Functions] [Modules]

DESCRIPTION

These functions manage callbacks in RexxDW.

6.4.1. CallbackManagement/DW_signal_connect [Functions]

[Top] [CallbackManagement] [Functions]

NAME

DW_signal_connect

SYNOPSIS

dw_signal_connect(Win, Signal, Function[, UserData1[, UserData2[, ...]]])

FUNCTION

Connects an event to a callback function.

ARGUMENTS

- Win - the window identifier to which the callback is assigned
- Signal - the event to be connected
- Function - the Rexx procedure that is called when the event fires
- UserData - (optional) user supplied data to pass to the Function

RESULT

No return result.

SEE ALSO

DW_signal_disconnect, DW_signal_disconnect_by_window, Callbacks

SOURCE

```
Call dw_signal_connect window, !REXXDW.!DW_CONFIGURE_EVENT, ,
   'configure_cb', 'fred', 'mary'
```

6.4.2. CallbackManagement/DW_signal_disconnect [Functions]

[Top] [CallbackManagement] [Functions]

NAME

DW_signal_disconnect

SYNOPSIS

dw_signal_disconnect(Win, Signal)

FUNCTION

Disconnects a specific event callback.

ARGUMENTS

- Win - the window identifier that has the callback assigned
- Signal - the event to be disconnected

RESULT

No return result.

SEE ALSO

DW_signal_connect, DW_signal_disconnect_by_window, Callbacks

SOURCE

```
Call dw_signal_connect window, !REXXDW.!DW_CONFIGURE_EVENT, ,
   'configure_cb', 'fred', 'mary'
...
Call dw_signal_disconnect window, !REXXDW.!DW_CONFIGURE_EVENT
```

6.4.3. CallbackManagement/DW_signal_disconnect_by_window [Functions]

[Top] [CallbackManagement] [Functions]

NAME

DW_signal_disconnect_by_window

SYNOPSIS

dw_signal_disconnect_by_window(Win)

FUNCTION

Disconnects all signals for a window.

ARGUMENTS

- Win - the window identifier that has the callbacks assigned

RESULT

No return result.

SEE ALSO

DW_signal_connect, DW_signal_disconnect, Callbacks

SOURCE

```
Call dw_signal_connect window, !REXXDW.!DW_CONFIGURE_EVENT, ,
   'configure_cb', 'fred', 'mary'
...
Call dw_signal_disconnect_by_window window
```

6.4.4. CallbackManagement/DW_timer_connect [Functions]

[Top] [CallbackManagement] [Functions]

NAME

DW_timer_connect

SYNOPSIS

timer = dw_timer_connect(Interval, Function[, UserData1[, UserData2[, ...]]])

FUNCTION

Creates a new timer callback. A timer callback will fire after the specified Interval and call the Rexx procedure Function with the optional arguments.

ARGUMENTS

- Interval - the interval in milliseconds that the timer will fire
- Function - the Rexx procedure that is called when the timer fires
- UserData* - optional user supplied data to pass to the Function

RESULT

A timer identifier.

SEE ALSO

DW_timer_disconnect, Callbacks

NOTES

There are two ways to stop a timer from firing. One is explicitly by calling DW_timer_disconnect() using the timer identifier returned from this function. The other way is to return 0 from the callback connected to the timer. Returning 1 from the callback will re-arm the timer.

SOURCE

```
timer = dw_timer_connect( 5000, 'timer_cb', 'fred', 'mary' )
```

6.4.5. CallbackManagement/DW_timer_disconnect [Functions]

[Top] [CallbackManagement] [Functions]

NAME

DW_timer_disconnect

SYNOPSIS

dw_timer_disconnect(Timer)

FUNCTION

Terminates the timer so the timer no longer files.

ARGUMENTS

- Timer - the timer identifier returned from a call to DW_timer_connect()

RESULT

No return result.

SEE ALSO

DW_timer_connect, Callbacks

SOURCE

```
timer = dw_timer_connect( 5000, 'timer_cb', 'fred', 'mary' )
...
Call dw_timer_disconnect timer
```

6.4.6. CallbackManagement/DW_callback_get_timestamp [Functions]

[Top] [CallbackManagement] [Functions]

NAME

DW_callback_get_timestamp

SYNOPSIS

timestamp = dw_callback_get_timestamp()

FUNCTION

Returns the timestamp (time_t value) of when the last callback fired.

ARGUMENTS

- None

RESULT

The timestamp of the last callback.

SEE ALSO

Callbacks

SOURCE

```
if dw_callback_get_timestamp() + 600 < time('T)' then say 'No activity in the application for 10
```

6.5. Functions/Browsing [Modules]

[Top] [Functions] [Modules]

DESCRIPTION

These functions provide browsing capabilties.

6.5.1. Browsing/DW_browse [Functions]

[Top] [Browsing] [Functions]

NAME

DW_browse

SYNOPSIS

rc = dw_browse(URL)

FUNCTION

Starts up the default Web browser as a new process with the specified URL. When using GTK, the browser started is "netscape". To use another browser set the environment variable DW_BROWSER to the executable to run.

ARGUMENTS

- URL - the Uniform Resource Locatr to open
- Padding - The number of pixels of padding to add around all sides of BoxToPack

RESULT

The return code from the attempt to execute the browser. Usually 0 indicates success and any other value an error.

SOURCE

```
...
rc = dw_browse( "http://rexxdw.sf.net" )
```

6.5.2. Browsing/DW_file_browse [Functions]

[Top] [Browsing] [Functions]

NAME

DW_file_browse

SYNOPSIS

filename = dw_file_browse(Title, DefaultDir, Extension, Flag)

FUNCTION

Creates a file open/save or directory browser dialog.

ARGUMENTS

- Title - the title of the dialog
- DefaultDir - the directory in which the dialog starts
- Ext - a file extension to filter the files displayed, or the file extension to save. Ignored for directory browser.
- Flag - indicates if the browse dialog is for a file open, file save or directory browse. One of !REXXDW.!DW_DIRECTORY_OPEN, !REXXDW.!DW_FILE_OPEN or !REXXDW.!DW_FILE_SAVE

RESULT

The fully qualified selected file/directory or blank if the dialog is cancelled.

SEE ALSO

FileDialogFlags

NOTES

On some platforms the current directory is changed when this function is called.

SOURCE

```
...
here = Directory()
dir = dw_file_browse( "Select Directory", "/home/mark", , ,
   !REXXDW.!DW_DIRECTORY_OPEN )
Say "Directory selected was:" dir
Call Directory( here )
```

6.6. Functions/ColourSupport [Modules]

[Top] [Functions] [Modules]

DESCRIPTION

These functions allow colours to be manipulated.

6.6.1. ColourSupport/DW_color_depth_get [Functions]

[Top] [ColourSupport] [Functions]

NAME

DW_color_depth_get

SYNOPSIS

depth = dw_color_depth_get()

FUNCTION

Returns the colour depth of the current screen.

ARGUMENTS

None

RESULT

The colour depth.

SOURCE

```
...
Say 'Colour depth is:' dw_color_depth_get()
```

6.6.2. ColourSupport/DW_rgb [Functions]

[Top] [ColourSupport] [Functions]

NAME

DW_rgb

SYNOPSIS

colour = dw_rgb(Red, Green, Blue)

FUNCTION

Mixes shades of red, green and blue to make an internal RexxDW colour.

ARGUMENTS

- Red - the amount of Red to use (0 to 255)
- Green - the amount of Green to use (0 to 255)
- Blue - the amount of Blue to use (0 to 255)

RESULT

An internal RexxDW colour.

SEE ALSO

DW_red_value, DW_green_value, DW_red_value

SOURCE

```
...
colour = dw_rgb( 45, 134, 200 )
```

6.6.3. ColourSupport/DW_red_value [Functions]

[Top] [ColourSupport] [Functions]

NAME

DW_red_value

SYNOPSIS

red = dw_red_value(Colour)

FUNCTION

Obtains the red component from a RexxDW colour.

ARGUMENTS

- Colour - the internal RexxDW colour containing a red component

RESULT

The red component of the colour.

SEE ALSO

DW_green_value, DW_blue_value, DW_rgb

SOURCE

```
...
Say 'The red component of DARKCYAN is:' ,
   dw_red_value( !REXXDW.!DW_CLR_DARKCYAN )
```

6.6.4. ColourSupport/DW_green_value [Functions]

[Top] [ColourSupport] [Functions]

NAME

DW_green_value

SYNOPSIS

green = dw_green_value(Colour)

FUNCTION

Obtains the green component from a RexxDW colour.

ARGUMENTS

- Colour - the internal RexxDW colour containing a green component

RESULT

The green component of the colour.

SEE ALSO

DW_red_value, DW_blue_value, DW_rgb

SOURCE

```
...
Say 'The green component of DARKCYAN is:' ,
   dw_green_value( !REXXDW.!DW_CLR_DARKCYAN )
```

6.6.5. ColourSupport/DW_blue_value [Functions]

[Top] [ColourSupport] [Functions]

NAME

DW_blue_value

SYNOPSIS

blue = dw_blue_value(Colour)

FUNCTION

Obtains the blue component from a RexxDW colour.

ARGUMENTS

- Colour - the internal RexxDW colour containing a blue component

RESULT

The blue component of the colour.

SEE ALSO

DW_red_value, DW_green_value, DW_rgb

SOURCE

```
...
Say 'The blue component of DARKCYAN is:' ,
   dw_blue_value( !REXXDW.!DW_CLR_DARKCYAN )
```

6.6.6. ColourSupport/DW_color_background_set [Functions]

[Top] [ColourSupport] [Functions]

NAME

DW_color_background_set

SYNOPSIS

dw_color_background_set(Colour)

FUNCTION

Sets the background colour of the specified window. Only applicable to render??? windows.

ARGUMENTS

- Colour - either one of the pre-defined colour constatnts, or a colour returned from a call to DW_rgb()

RESULT

No return result.

SEE ALSO

Colours, DW_color_foreground_set, DW_rgb

SOURCE

```
...
Call dw_color_background_set( renderbox, dw_rgb( 10, 45, 123 ) )
```

6.6.7. ColourSupport/DW_color_foreground_set [Functions]

[Top] [ColourSupport] [Functions]

NAME

DW_color_foreground_set

SYNOPSIS

dw_color_foreground_set(Colour)

FUNCTION

Sets the foreground colour of the specified window. Only applicable to render??? windows.

ARGUMENTS

- Colour - either one of the pre-defined colour constatnts, or a colour returned from a call to DW_rgb()

RESULT

No return result.

SEE ALSO

Colours, DW_color_background_set, DW_rgb

SOURCE

```
...
Call dw_color_foreground_set( renderbox, dw_rgb( 10, 45, 123 ) )
```

6.6.8. ColourSupport/DW_color_choose [Functions]

[Top] [ColourSupport] [Functions]

NAME

DW_color_choose

SYNOPSIS

color = dw_color_choose(InitialColour)

FUNCTION

Displays a system dialog for selecting a color.

ARGUMENTS

- InitialColour - either one of the pre-defined colour constatnts, or a colour returned from a call to DW_rgb()

RESULT

The selected color, or the same value as InitialColor if the user cancels.

SEE ALSO

Colours, DW_color_background_set, DW_rgb

SOURCE

```
...
newcolor = dw_color_choose( dw_rgb( 10, 45, 123 )
```

6.7. Functions/FontSupport [Modules]

[Top] [Functions] [Modules]

DESCRIPTION

These functions support manipulation of fonts.

6.7.1. FontSupport/DW_font_choose [Functions]

[Top] [FontSupport] [Functions]

NAME

DW_font_choose

SYNOPSIS

font = dw_font_choose([CurrentFont])

FUNCTION

Displays a font chooser dialog

ARGUMENTS

- CurrentFont - the current font to set as the basis for choosing a font

RESULT

The selected font or blank if no font chosen

SOURCE

```
font = dw_font_choose( '8.Monaco' )
```

6.7.2. FontSupport/DW_font_set_default [Functions]

[Top] [FontSupport] [Functions]

NAME

DW_font_set_default

SYNOPSIS

dw_font_set_default(Font)

FUNCTION

Sets the default font for all widgets with a text component.

ARGUMENTS

- Font - the font to be used as the default in Dynamic Windows format

RESULT

None

SOURCE

```
Call dw_font_set_default, '8.Monaco'
```

6.7.3. FontSupport/DW_font_text_extents_get [Functions]

[Top] [FontSupport] [Functions]

NAME

DW_font_text_extents_get

SYNOPSIS

Width Height = dw_font_text_extents_get(Win, Pixmap, Text)

FUNCTION

Returns the width and height of the Text in the current font.

ARGUMENTS

- Win - the window identifier returned from DW_container_new()
- Pixmap - the pixmap identifier returned from DW_pixmap_new()
- Text - the string to measure the font width and height

RESULT

Width and Height are returned as two words. PARSE VALUE is the easiest way to obtain both values.

NOTES

When calculating the size of the maximum width and height of a font, use a value like 'g(' for Text to ensure the maximum height of the Text can be calculated. Only one of Win or Pixmap is required. The other argument should be set to 0.

SOURCE

```
win = dw_render_new( 0 )
```

```
Call dw_window_set_font win, myfont
Parse Value dw_font_text_extents_get( win, 0, 'g(' ) With width height
```

6.8. Functions/ModuleSupport [Modules]

[Top] [Functions] [Modules]

DESCRIPTION

These functions support the loading and unloading of modules in DLLs. Not applicable to RexxDW.

6.9. Functions/MutexSupport [Modules]

[Top] [Functions] [Modules]

DESCRIPTION

These functions support the processing of semaphores. Not applicable to RexxDW.

6.10. Functions/EventSupport [Modules]

[Top] [Functions] [Modules]

DESCRIPTION

These functions support events. Not applicable to RexxDW.

6.11. Functions/ThreadSupport [Modules]

[Top] [Functions] [Modules]

DESCRIPTION

These functions support threads. Not applicable to RexxDW.

6.12. Functions/PointerPosition [Modules]

[Top] [Functions] [Modules]

DESCRIPTION

These functions allow positioning of the mouse pointer.

6.12.1. PointerPosition/DW_pointer_set_pos [Functions]

[Top] [PointerPosition] [Functions]

NAME

DW_pointer_set_pos

SYNOPSIS

dw_pointer_set_pos(X, Y)

FUNCTION

Moves the mouse pointer to the specified X and Y coordinates.

ARGUMENTS

- X - the X coordinate of the mouse pointer
- Y - the Y coordinate of the mouse pointer

RESULT

No return result.

SEE ALSO

DW_pointer_get_pos

NOTES

The coordinates are absolute screen positions, with 0/0 being the the top left corner of the screen. (Windows!!)

SOURCE

```
...
Call dw_pointer_set_pos 100, 130
```

6.12.2. PointerPosition/DW_pointer_get_pos [Functions]

[Top] [PointerPosition] [Functions]

NAME

DW_pointer_get_pos

SYNOPSIS

X Y = dw_pointer_get_pos( )

FUNCTION

Returns the X and Y coordinates of the mouse pointer.

ARGUMENTS

None

RESULT

- X and Y are returned as two words. PARSE VALUE is the easiest

way to obtain both values.

SEE ALSO

DW_pointer_set_pos

NOTES

The coordinates are absolute screen positions, with 0/0 being the the top left corner of the screen. (Windows!!)

SOURCE

```
...
Parse Value dw_pointer_get_pos( ) With x y
Say 'The mouse pointer is at' x'/'y
```

6.13. Functions/Utility [Modules]

[Top] [Functions] [Modules]

DESCRIPTION

These functions are grouped together becasue they don't fit into any other group.

6.13.1. Utility/DW_app_dir [Functions]

[Top] [Utility] [Functions]

NAME

DW_app_dir

SYNOPSIS

dir = dw_app_dir( )

FUNCTION

Attempts to return the applications data directory.

ARGUMENTS

None

RESULT

The application data directory.

SOURCE

```
...
dir = dw_app_dir()
Say 'App data dir is:' dir
```

6.13.2. Utility/DW_beep [Functions]

[Top] [Utility] [Functions]

NAME

DW_beep

SYNOPSIS

dw_beep(Frequency, Duration)

FUNCTION

Makes a sound of the given Frequency for the given Duration.

ARGUMENTS

- Frequency - the frequency of the sound

- Duration - the duration of the sound in milliseconds

RESULT

No return result.

SOURCE

```
...
Call dw_beep(5000, 1000)
```

6.13.3. Utility/DW_debug [Functions]

[Top] [Utility] [Functions]

NAME

DW_debug

SYNOPSIS

dw_debug(String)

FUNCTION

Displays the supplied String to the system's debugging location. This is usually the standard error stream.

ARGUMENTS

- String - the string to display

RESULT

No return result.

SOURCE

```
...
Call dw_debug('This is a debug string')
```

6.13.4. Utility/DW_environment_query [Functions]

[Top] [Utility] [Functions]

NAME

DW_environment_query

SYNOPSIS

dw_environment_query(Stem)

FUNCTION

Returns lots of information about the platform and Dynamic Windows in an array.

ARGUMENTS

- Stem - stem (with trailing period) to place the results

RESULT

No return result. Value of array items:

```
- Stem.0  - 10 (number of items in array)
- Stem.1  - Operating System Name
- Stem.2  - Operating System Build Date
- Stem.3  - Operating System Build Time
- Stem.4  - Operating System Major Version
- Stem.5  - Operating System Minor Version
- Stem.6  - Operating System Major Build Number
- Stem.7  - Operating System Minor Build Number
- Stem.8  - Dynamic Windows Major Version
- Stem.9  - Dynamic Windows Minor Version
- -Stem.10 - Dynamic Windows Sub Version
```

SOURCE

```
Call dw_environment_query( 'stem.' )
Say 'OS name is:' stem.1
```

6.13.5. Utility/DW_user_dir [Functions]

[Top] [Utility] [Functions]

NAME

DW_user_dir

SYNOPSIS

dir = dw_user_dir( )

FUNCTION

Returns the user's "home" directory. If the environment variable HOME is set, this is returned, otherwise "C:\" is returned.

ARGUMENTS

None

RESULT

The user's home directory.

SOURCE

```
Say 'home is' dw_user_dir()
```

6.13.6. Utility/DW_screen_height [Functions]

[Top] [Utility] [Functions]

NAME

DW_screen_height

SYNOPSIS

height = dw_screen_height( )

FUNCTION

Returns the height of the physical screen in pixels.

ARGUMENTS

None

RESULT

The screen height

SEE ALSO

DW_screen_width

SOURCE

```
Say 'The height of the screen is' dw_screen_height()
```

6.13.7. Utility/DW_screen_width [Functions]

[Top] [Utility] [Functions]

NAME

DW_screen_width

SYNOPSIS

width = dw_screen_width( )

FUNCTION

Returns the width of the physical screen in pixels.

ARGUMENTS

None

RESULT

The screen width

SEE ALSO

DW_screen_height

SOURCE

```
Say 'The width of the screen is' dw_screen_width()
```

6.13.8. Utility/DW_clipboard_get_text [Functions]

[Top] [Utility] [Functions]

NAME

DW_clipboard_get_text

SYNOPSIS

str = dw_clipboard_get_text( )

FUNCTION

Returns the textual contents of the default clipboard.

ARGUMENTS

None

RESULT

The contents of the default clipboard if it can be converted to text.

SEE ALSO

DW_clipboard_set_text

SOURCE

```
Say 'The clipboard contains' dw_clipboard_get_text()
```

6.13.9. Utility/DW_clipboard_set_text [Functions]

[Top] [Utility] [Functions]

NAME

DW_clipboard_set_text

SYNOPSIS

dw_clipboard_set_text(Str)

FUNCTION

Sets the contents of the default clipboard to the specified text.

ARGUMENTS

- Str - new clipboard contents

RESULT

No return result.

SEE ALSO

DW_clipboard_get_text

SOURCE

```
Call dw_clipboard_set_text 'The clipboard contains this.'
```

6.13.10. Utility/DW_or [Functions]

[Top] [Utility] [Functions]

NAME

DW_or

SYNOPSIS

val = dw_or(Val1, Val2[,Val3...]])

FUNCTION

Logically "or" multiple values together.

ARGUMENTS

- Val1 - value of first numeric value to logically "or"
- Val2 - value of second numeric value to logically "or"
- ... - optional other numeric values to logically "or"

RESULT

The combined value of all arguments logically "or"ed together.

SEE ALSO

DW_and

SOURCE

```
...
style = dw_or( !REXXDW.!DW_FCF_SYSMENU, !REXXDW.!DW_FCF_TITLEBAR, ,
   !REXXDW.!DW_FCF_SHELLPOSITION, !REXXDW.!DW_FCF_TASKLIST, ,
   !REXXDW.!DW_FCF_DLGBORDER, !REXXDW.!DW_FCF_SIZEBORDER, ,
   !REXXDW.!DW_FCF_MINMAX )
win = dw_window_new( !REXXDW.!DW_DESKTOP, 'Window on desktop', style )
```

6.13.11. Utility/DW_and [Functions]

[Top] [Utility] [Functions]

NAME

DW_and

SYNOPSIS

val = dw_and(Val1, Val2[,Val3...]])

FUNCTION

Logically "and" multiple values together.

ARGUMENTS

- Val1 - value of first numeric value to logically "and"

- Val2 - value of second numeric value to logically "and"
- ... - optional other numeric values to logically "and"

RESULT

The combined value of all arguments logically "and"ed together.

SEE ALSO

DW_or

SOURCE

```
...
val = dw_and(10, 512345)
```

6.13.12. Utility/DW_flush [Functions]

[Top] [Utility] [Functions]

NAME

DW_flush

SYNOPSIS

dw_flush()

FUNCTION

Ensures that any changes drawn to the screen are immediately visible.

ARGUMENTS

None

RESULT

No return result.

SOURCE

```
...
Call dw_pixmap_bitblt win, , 0, 0, width, height, , pixmap, x, 0
Call dw_flush()
```

6.14. Functions/PackageManagement [Modules]

[Top] [Functions] [Modules]

DESCRIPTION

These functions are common to most Rexx external function packages.

6.14.1. PackageManagement/DW_loadfuncs [Functions]

[Top] [PackageManagement] [Functions]

NAME

DW_loadfuncs

SYNOPSIS

rcode = dw_loadfuncs()

FUNCTION

Loads all other RexxDW external functions

ARGUMENTS

None

RESULT

0 in all cases

SEE ALSO

DW_dropfuncs

NOTES

Under OS/2, **DW_loadfuncs**() will set the current process to a PM application to allow interaction with the OS/2 message queue. DW_dropfuncs() sets the process back to the type of application it was started as. If you run a Rexx/DW program as a .CMD file from a CMD.EXE window, and DO NOT call DW_dropfuncs(), your CMD.EXE window will be unusable!

6.14.2. PackageManagement/DW_dropfuncs [Functions]

[Top] [PackageManagement] [Functions]

NAME

DW_dropfuncs

SYNOPSIS

rcode = dw_dropfuncs(["UNLOAD"])

FUNCTION

Cleans up RexxDW environment and optionally will drop the external functions.

ARGUMENTS

- UNLOAD - causes the external functions to be dropped.

RESULT

0 in all cases

SEE ALSO

DW_loadfuncs

NOTES

Under OS/2, DW_loadfuncs() will set the current process to a PM application to allow interaction with the OS/2 message queue. **DW_dropfuncs**() sets the process back to the type of application it was started as. If you run a Rexx/DW program as a .CMD file from a CMD.EXE window, and DO NOT call **DW_dropfuncs**(), your CMD.EXE window will be unusable!

6.14.3. PackageManagement/DW_variable [Functions]

[Top] [PackageManagement] [Functions]

NAME

DW_variable

SYNOPSIS

rcode = dw_variable(Variable [,NewValue])

FUNCTION

Get or set an internal RexxDW variable.

ARGUMENTS

- Variable - name of the variable to get or set. See NOTES for
- NewValue - the new value of "Variable", if the variable is settable

RESULT

When setting a variable, then 0 if success, any other value is an error When getting a variable, then the value of the variable is returned.

NOTES

The "Variable" argument can be one of:

```
DEBUG (settable)
   0 - no debugging
   1 - all Rexx variables set by RexxDW are displayed as they are set
   2 - all RexxDW functions are traced on entry with argument values and
       on exit with the return value
   4 - all internal RexxDW functions are traced with their arguments
       (really only useful for developers)
   The values can be added together for a combination of the above details.
DEBUGFILE (settable)
   Where any debugging output is written. By default this goes to
   the system's error stream; usually 'stderr'.
CONSTANTPREFIX (settable)
   The variable name prefix for all RexxDW constants. By default this is
   '!REXXDW.!'. If you change this, it is useful to make the prefix result
   in stemmed variables; this makes it far easier to EXPOSE these constants.
VERSION (readonly)
   The full version details of RexxDW in the format:
   package version version_date
   Where:
      package      - the string 'rexxdw'
      version      - package version in n.n format; eg 1.0
      version_date - date package was released in DATE('N') format
```

SOURCE

```
...
Say 'We are running at debug level:' dw_variable( 'DEBUG' )
```

6.14.4. PackageManagement/DW_QueryFunction [Functions]

[Top] [PackageManagement] [Functions]

NAME

DW_QueryFunction

SYNOPSIS

rcode = dw_QueryFunction(FunctionName|ResultArray[, Option])

FUNCTION

Populates an array of all functions supplied by this package depending on Option

ARGUMENTS

- FunctionName - the name of a function to query (no trailing period)
- ResultArray - the stem (trailing period) in which the list of functions is returned
- Option - one of 'R' (the default) for "registered" functions or 'A' for "available" functions

RESULT

0 if successful or 1 if the FunctionName is not found

NOTES

To determine if a FunctionName can be executed in your code, pass the function name as the first argument, and 'R' as the second. If the function can be called the function returns 0, otherwise it returns 1