

#\$+K!

Gizmos: a library of enhanced controls

by Julian Smart and others

January 5th 2002

Contents

[Copyright notice](#)

[Introduction](#)

[Alphabetical class reference](#)

[Classes by category](#)

[Topic overviews](#)

[References](#)

C
ontents
C
ontents
b
rowse00001
K
Contents
D
isableButton("Up")

Copyright notice

The licence is the wxWindows Licence.

Introduction

What is the Gizmos library?

Introduction
Introduction
Browse00003
Introduction
DisableButton("Up")

`##+K!` Alphabetical class reference

[wxDynamicSashSplitEvent](#)
[wxDynamicSashUnifyEvent](#)
[wxDynamicSashWindow](#)
[wxEditableListBox](#)
[wxLEDNumberCtrl](#)
[wxMultiCellCanvas](#)
[wxMultiCellItemHandle](#)
[wxMultiCellSizer](#)
[wxRemotelyScrolledTreeCtrl](#)
[wxSplitterScrolledWindow](#)
[wxThinSplitterWindow](#)
[wxTreeCompanionWindow](#)

^A Alphabetical class reference

^C lassref

^b rowse00005

^K Alphabetical class reference

^D isableButton("Up")

Classes by category

A classification of Gizmos classes by category.

Topic overviews

This chapter contains a selection of topic overviews, first things first:

[Notes on using the reference](#)

Topic overviews
overviews
rowse00105
Topic overviews
isableButton("Up")

References

References
Bibliography
Browse00107
References
DisableButton("Up")

\$#+K!What is the Gizmos library?

This manual describes a class library with a miscellany of useful user interface classes.

W hat is the Gizmos library?

t opic1

b rowse00004

K What is the Gizmos library?

E nableButton("Up");ChangeButtonBinding("Up", "JumpId(^gizmos.hlp',`introduction')")

wxDynamicSashSplitEvent

wxDynamicSashSplitEvents are sent to your view by wxDynamicSashWindow whenever your view is being split by the user. It is your responsibility to handle this event by creating a new view window as a child of the wxDynamicSashWindow. wxDynamicSashWindow will automatically re-parent it to the proper place in its window hierarchy.

Derived from

[wxCommandEvent](#)

Data structures

Members

[wxDynamicSashSplitEvent::wxDynamicSashSplitEvent](#)

[wxDynamicSashSplitEvent::Clone](#)

^wxDynamicSashSplitEvent

^wxdynamicssashsplitevent

^browse00006

^K wxDynamicSashSplitEvent

^EnableButton("Up");ChangeButtonBinding("Up", "JumpId(^gizmos.hlp', `classref)")

wxDynamicSashUnifyEvent

wxDynamicSashUnifyEvents are sent to your view by wxDynamicSashWindow whenever the sash which splits your view and its sibling is being reunified such that your view is expanding to replace its sibling. You needn't do anything with this event if you are allowing wxDynamicSashWindow to manage your view's scrollbars, but it is useful if you are managing the scrollbars yourself so that you can keep the scrollbars' event handlers connected to your view's event handler class.

Derived from

[wxCommandEvent](#)

Data structures

Members

[wxDynamicSashUnifyEvent::wxDynamicSashUnifyEvent](#)

[wxDynamicSashUnifyEvent::Clone](#)

^wxDynamicSashUnifyEvent

^wxdynamicsshunifyevent

^browse00009

^K wxDynamicSashUnifyEvent

^EnableButton("Up");ChangeButtonBinding("Up", "JumpId(^gizmos.hlp', `classref)")

wxDynamicSashWindow

wxDynamicSashWindow. See above.

Derived from

[wxWindow](#)

Data structures

Members

[wxDynamicSashWindow::wxDynamicSashWindow](#)
[wxDynamicSashWindow::~~wxDynamicSashWindow](#)
[wxDynamicSashWindow::AddChild](#)
[wxDynamicSashWindow::Create](#)
[wxDynamicSashWindow::GetHScrollBar](#)
[wxDynamicSashWindow::GetVScrollBar](#)

^wxDynamicSashWindow

^wxdynamicsshwindow

^browse00012

^K wxDynamicSashWindow

^EnableButton("Up");ChangeButtonBinding("Up", "JumpId(^gizmos.hlp', `classref)")

\$#+K! **wxEitableListBox**

This class provides a composite control that lets the user easily enter list of strings

Derived from

[wxPanel](#)

Data structures

Members

[wxEditableListBox::wxEditableListBox](#)
[wxEditableListBox::GetStrings](#)
[wxEditableListBox::OnDellItem](#)
[wxEditableListBox::OnDownItem](#)
[wxEditableListBox::OnEditItem](#)
[wxEditableListBox::OnEndLabelEdit](#)
[wxEditableListBox::OnItemSelected](#)
[wxEditableListBox::OnNewItem](#)
[wxEditableListBox::OnUpItem](#)
[wxEditableListBox::SetStrings](#)

^wxEditableListBox

^wxeditablelistbox

^browse00019

^K wxEditableListBox

^EnableButton("Up");ChangeButtonBinding("Up", "JumpId(^gizmos.hlp', `classref)")

\$#+K! **wxLEDNumberCtrl**

wxLEDNumberCtrl

Derived from

wxControl

Data structures

Members

wxLEDNumberCtrl::wxLEDNumberCtrl

wxLEDNumberCtrl::Create

wxLEDNumberCtrl::GetAlignment

wxLEDNumberCtrl::GetDrawFaded

wxLEDNumberCtrl::GetValue

wxLEDNumberCtrl::SetAlignment

wxLEDNumberCtrl::SetDrawFaded

wxLEDNumberCtrl::SetValue

^wxLEDNumberCtrl

^wxlednumberctrl

^browse00030

^K wxLEDNumberCtrl

^EnableButton("Up");ChangeButtonBinding("Up", "JumpId(^gizmos.hlp', `classref)")

`##+K!` **wxMultiCellCanvas**

wxCell is used internally, so we don't need to declare it here

wxMultiCellCanvas

Derived from

wxFlexGridSizer

Data structures

Members

wxMultiCellCanvas::wxMultiCellCanvas
wxMultiCellCanvas::Add
wxMultiCellCanvas::CalculateConstraints
wxMultiCellCanvas::MaxCols
wxMultiCellCanvas::MaxRows
wxMultiCellCanvas::Resize
wxMultiCellCanvas::SetMinCellSize

`w` wxMultiCellCanvas
`w` wxmulticellcanvas
`b` rowse00039
`K` wxMultiCellCanvas
`E` nableButton("Up");ChangeButtonBinding("Up", "JumpId(^gizmos.hlp', `classref)")

\$#+K! **wxMultiCellItemHandle**

classes

wxMultiCellItemHandle

Derived from

wxObject

Data structures

Members

wxMultiCellItemHandle::wxMultiCellItemHandle

wxMultiCellItemHandle::GetAlignment

wxMultiCellItemHandle::GetColumn

wxMultiCellItemHandle::GetHeight

wxMultiCellItemHandle::GetLocalSize

wxMultiCellItemHandle::GetRow

wxMultiCellItemHandle::GetStyle

wxMultiCellItemHandle::GetWeight

wxMultiCellItemHandle::GetWidth

^wxMultiCellItemHandle

^wxmulticellitemhandle

^browse00047

^K wxMultiCellItemHandle

^EnableButton("Up");ChangeButtonBinding("Up", "JumpId(^gizmos.hlp', `classref)")

\$#+K! **wxMultiCellSizer**

wxMultiCellSizer

Derived from

wxSizer

Data structures

Members

wxMultiCellSizer::wxMultiCellSizer
wxMultiCellSizer::~~wxMultiCellSizer
wxMultiCellSizer::CalcMin
wxMultiCellSizer::EnableGridLines
wxMultiCellSizer::OnPaint
wxMultiCellSizer::RecalcSizes
wxMultiCellSizer::SetColumnWidth
wxMultiCellSizer::SetDefaultCellSize
wxMultiCellSizer::SetGridPen
wxMultiCellSizer::SetRowHeight

^wxMultiCellSizer

^wxmulticellsizer

^browse00057

^K wxMultiCellSizer

^EnableButton("Up");ChangeButtonBinding("Up", "JumpId(^gizmos.hlp', `classref)")

wxRemotelyScrolledTreeCtrl

wxRemotelyScrolledTreeCtrl This tree control disables its vertical scrollbar and catches scroll events passed by a scrolled window higher in the hierarchy. It also updates the scrolled window vertical scrollbar as appropriate. **Derived from**

[wxTreeCtrl](#)

Data structures

Members

[wxRemotelyScrolledTreeCtrl::wxRemotelyScrolledTreeCtrl](#)
[wxRemotelyScrolledTreeCtrl::~~wxRemotelyScrolledTreeCtrl](#)
[wxRemotelyScrolledTreeCtrl::AdjustRemoteScrollbars](#)
[wxRemotelyScrolledTreeCtrl::CalcTreeSize](#)
[wxRemotelyScrolledTreeCtrl::GetCompanionWindow](#)
[wxRemotelyScrolledTreeCtrl::GetScrollPos](#)
[wxRemotelyScrolledTreeCtrl::GetScrolledWindow](#)
[wxRemotelyScrolledTreeCtrl::GetViewStart](#)
[wxRemotelyScrolledTreeCtrl::HideVScrollbar](#)
[wxRemotelyScrolledTreeCtrl::OnExpand](#)
[wxRemotelyScrolledTreeCtrl::OnPaint](#)
[wxRemotelyScrolledTreeCtrl::OnScroll](#)
[wxRemotelyScrolledTreeCtrl::OnSize](#)
[wxRemotelyScrolledTreeCtrl::PrepareDC](#)
[wxRemotelyScrolledTreeCtrl::ScrollToLine](#)
[wxRemotelyScrolledTreeCtrl::SetCompanionWindow](#)
[wxRemotelyScrolledTreeCtrl::SetScrollbars](#)

^w[wxRemotelyScrolledTreeCtrl](#)

^w[xremotelyscrolledtreectrl](#)

^b[rowse00068](#)

^K [wxRemotelyScrolledTreeCtrl](#)

^E[nableButton\("Up"\);ChangeButtonBinding\("Up", "JumpId\(^gizmos.hlp', `classref'\)"\)](#)

\$#+K! **wxSplitterScrolledWindow**

wxSplitterScrolledWindow This scrolled window is aware of the fact that one of its children is a splitter window. It passes on its scroll events (after some processing) to both splitter children for them to scroll appropriately. **Derived from**

wxScrolledWindow

Data structures

Members

wxSplitterScrolledWindow::wxSplitterScrolledWindow

wxSplitterScrolledWindow::OnScroll

wxSplitterScrolledWindow::OnSize

wxSplitterScrolledWindow

wxsplitterscrolledwindow

browse00086

K wxSplitterScrolledWindow

EnableButton("Up");ChangeButtonBinding("Up", "JumpId(^gizmos.hlp', `classref)")

`wxThinSplitterWindow`

`wxThinSplitterWindow` Implements a splitter with a less obvious sash than the usual one. **Derived from**

[wxSplitterWindow](#)

Data structures

Members

[wxThinSplitterWindow::wxThinSplitterWindow](#)

[wxThinSplitterWindow::DrawSash](#)

[wxThinSplitterWindow::OnSize](#)

[wxThinSplitterWindow::SashHitTest](#)

[wxThinSplitterWindow::SizeWindows](#)

`wxThinSplitterWindow`

`wxthin splitterwindow`

`rowse00090`

`wxThinSplitterWindow`

`nableButton("Up");ChangeButtonBinding("Up", "JumpId(^gizmos.hlp', `classref)")`

\$#+K! **wxTreeCompanionWindow**

wxTreeCompanionWindow A window displaying values associated with tree control items. **Derived from**

wxWindow

Data structures

Members

wxTreeCompanionWindow::wxTreeCompanionWindow

wxTreeCompanionWindow::DrawItem

wxTreeCompanionWindow::GetTreeCtrl

wxTreeCompanionWindow::OnExpand

wxTreeCompanionWindow::OnPaint

wxTreeCompanionWindow::OnScroll

wxTreeCompanionWindow::SetTreeCtrl

^wxTreeCompanionWindow

^wxtreecompanionwindow

^browse00096

^K wxTreeCompanionWindow

^EnableButton("Up");ChangeButtonBinding("Up", "JumpId(^gizmos.hlp', `classref)")

Notes on using the reference

In the descriptions of the wxWindows classes and their member functions, note that descriptions of inherited member functions are not duplicated in derived classes unless their behaviour is different. So in using a class such as wxScrolledWindow, be aware that wxWindow functions may be relevant.

Note also that arguments with default values may be omitted from a function call, for brevity. Size and position arguments may usually be given a value of -1 (the default), in which case wxWindows will choose a suitable value.

Most strings are returned as wxString objects. However, for remaining char * return values, the strings are allocated and deallocated by wxWindows. Therefore, return values should always be copied for long-term use, especially since the same buffer is often used by wxWindows.

The member functions are given in alphabetical order except for constructors and destructors which appear first.

Notes on using the reference

referencenotes

browse00106

Notes on using the reference

enableButton("Up");ChangeButtonBinding("Up", "JumpId(^gizmos.hlp', `overviews')")

`wxDynamicSashSplitEvent::wxDynamicSashSplitEvent`

`wxDynamicSashSplitEvent(const wxDynamicSashSplitEvent& event)`^K

`wxDynamicSashSplitEvent(wxObject* target)`^K

`wxDynamicSashSplitEvent()`^K

`wxDynamicSashSplitEvent::wxDynamicSashSplitEvent`

`wxdynamicsashspliteventwxdynamicsashsplitevent`

`rowse00007`

`wxDynamicSashSplitEvent wxDynamicSashSplitEvent`

`enableButton("Up");ChangeButtonBinding("Up", "JumpId(`gizmos.hlp', `wxdynamicsashsplitevent')")`

`wxDynamicSashSplitEvent`

`wxDynamicSashSplitEvent`

`wxDynamicSashSplitEvent`

`$#+KKl wxDynamicSashSplitEvent::Clone`

`wxEvent* Clone() const`

`wxDynamicSashSplitEvent::Clone`

`wxdynamicsashspliteventclone`

`browse00008`

`K wxDynamicSashSplitEvent Clone`

`K Clone`

`EnableButton("Up");ChangeButtonBinding("Up", "JumpId(^gizmos.hlp', `wxdynamicsashsplitevent')")`

$\$#+K!$ wxDynamicSashUnifyEvent::wxDynamicSashUnifyEvent

wxDynamicSashUnifyEvent(const wxDynamicSashUnifyEvent& *event*)^K

wxDynamicSashUnifyEvent(wxObject* *target*)^K

wxDynamicSashUnifyEvent()^K

w wxDynamicSashUnifyEvent::wxDynamicSashUnifyEvent
 w wxdynamicsashunifyevent wxdynamicsashunifyevent
 b rowse00010
 K wxDynamicSashUnifyEvent wxDynamicSashUnifyEvent
 E nableButton("Up");ChangeButtonBinding("Up", "JumpId(`gizmos.hlp', ` wxdynamicsashunifyevent')")
 K wxDynamicSashUnifyEvent
 K wxDynamicSashUnifyEvent
 K wxDynamicSashUnifyEvent

`$#+KKl wxDynamicSashUnifyEvent::Clone`

`wxEvent* Clone() const`

`wxDynamicSashUnifyEvent::Clone`

`wxdynamicsashunifyeventclone`

`browse00011`

`K wxDynamicSashUnifyEvent Clone`

`K Clone`

`EnableButton("Up");ChangeButtonBinding("Up", "JumpId(^gizmos.hlp', `wxdynamicsashunifyevent')")`

^{\$#+K!}**wxDynamicSashWindow::wxDynamicSashWindow**

wxDynamicSashWindow(**wxWindow*** *parent*, **wxWindowID** *id*, **const wxPoint&** *pos*
= *wxDefaultPosition*, **const wxSize&** *size* = *wxDefaultSize*, **long style** =
wxCLIP_CHILDREN | wxDS_MANAGE_SCROLLBARS | wxDS_DRAG_CORNER,
const wxString& *name* = "dynamicSashWindow")^K

wxDynamicSashWindow()^K

^wxDynamicSashWindow::wxDynamicSashWindow
^wxdynamicssashwindowwxdynamicssashwindow
^browse00013
^K wxDynamicSashWindow wxDynamicSashWindow
^EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`gizmos.hlp', `wxdynamicssashwindow')")
^K wxDynamicSashWindow
^K wxDynamicSashWindow

$\$#+K!$ wxDynamicSashWindow::~wxDynamicSashWindow

\sim wxDynamicSashWindow()^K

w wxDynamicSashWindow::~wxDynamicSashWindow
 w wxdynamicsashwindowdtor
 b rowse00014
 K wxDynamicSashWindow \sim wxDynamicSashWindow
 E nableButton("Up");ChangeButtonBinding("Up", "JumpId(^ gizmos.hlp', `wxdynamicsashwindow')")
 K \sim wxDynamicSashWindow

`$#+K!wxDynamicSashWindow::AddChild`

`void AddChild(wxWindowBase* child)K`

This is overloaded from `wxWindowBase`. It's not here for you to call directly.

`wxDynamicSashWindow::AddChild`
`wxdynamicsashwindowaddchild`
`rowse00015`
`K wxDynamicSashWindow AddChild`
`E nableButton("Up");ChangeButtonBinding("Up", "JumpId(^gizmos.hlp', `wxdynamicsashwindow')")`
`K AddChild`

`$#+K! wxDynamicSashWindow::Create`

**`bool Create(wxWindow* parent, wxWindowID id, const wxPoint& pos =
wxDefaultPosition, const wxSize& size = wxDefaultSize, long style =
wxCLIP_CHILDREN | wxDS_MANAGE_SCROLLBARS | wxDS_DRAG_CORNER,
const wxString& name = "dynamicSashWindow")K`**

^wxDynamicSashWindow::Create
^wxdynamicssashwindowcreate
^browse00016
^K wxDynamicSashWindow Create
^EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`gizmos.hlp', `wxdynamicssashwindow')")
^K Create

\$#+KKl wxDynamicSashWindow::GetHScrollBar

wxScrollBar* GetHScrollBar(const wxWindow* *child*) const

wxDynamicSashWindow::GetHScrollBar

wxdynamicsashwindowgethscrollbar

rowse00017

K wxDynamicSashWindow GetHScrollBar

K GetHScrollBar

E nableButton("Up");ChangeButtonBinding("Up", "JumpId(^ gizmos.hlp', `wxdynamicsashwindow')")

`$#+KKl wxDynamicSashWindow::GetVScrollBar`

`wxScrollBar* GetVScrollBar(const wxWindow* child) const`

`w_xDynamicSashWindow::GetVScrollBar`

`w_xdynamicsashwindowgetvscrollbar`

`browse00018`

`K wxDynamicSashWindow GetVScrollBar`

`K GetVScrollBar`

`E nableButton("Up");ChangeButtonBinding("Up", "JumpId(^gizmos.hlp', `wxdynamicsashwindow')")`

`$#+K! wxEditableListBox::wxEditableListBox`

**`wxEditableListBox(wxWindow* parent, wxWindowID id, const wxString& label,
const wxPoint& pos = wxDefaultPosition, const wxSize& size = wxDefaultSize, const
wxString& name = wxT("editableListBox"))K`**

^w`xEditableListBox::wxEditableListBox`
^w`xeditablelistboxwxeditablelistbox`
^b`rowse00020`
^K`wxEditableListBox wxEditableListBox`
^E`nableButton("Up");ChangeButtonBinding("Up", "JumpId(^gizmos.hlp', `wxeditablelistbox')")`
^K`wxEditableListBox`

`$#+K! wxEditableListBox::GetStrings`

`void GetStrings(wxArrayString& strings)K`

^wxEditableListBox::GetStrings
^wxeditablelistboxgetstrings
^browse00021
^K wxEditableListBox GetStrings
^EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`gizmos.hlp', `wxeditablelistbox')")
^K GetStrings

`$#+K! wxEditableListBox::OnDelItem`

`void OnDelItem(wxCommandEvent& event)K`

^wxEditableListBox::OnDelItem
^wxeditablelistboxondelitem
^browse00022
^K wxEditableListBox OnDelItem
^EnableButton("Up");ChangeButtonBinding("Up", "JumpId(^gizmos.hlp', `wxeditablelistbox')")
^K OnDelItem

`$#+K!wxEditableListBox::OnDownItem`

`void OnDownItem(wxCommandEvent& event)K`

`wxEditableListBox::OnDownItem`
`wxeditablelistboxondownitem`
`browse00023`
`K wxEditableListBox OnDownItem`
`EnableButton("Up");ChangeButtonBinding("Up", "JumpId(^gizmos.hlp', `wxeditablelistbox')")`
`K OnDownItem`

`$#+K! wxEditableListBox::OnEditItem`

`void OnEditItem(wxCommandEvent& event)K`

^wxEditableListBox::OnEditItem
^wxeditablelistboxonedititem
^browse00024
^K wxEditableListBox OnEditItem
^EnableButton("Up");ChangeButtonBinding("Up", "JumpId(^ gizmos.hlp', `wxeditablelistbox')")
^K OnEditItem

`$#+K!wxEditableListBox::OnEndLabelEdit`

`void OnEndLabelEdit(wxListEvent& event)K`

^wxEditableListBox::OnEndLabelEdit
^wxeditablelistboxonendlabeledit
^browse00025
^K wxEditableListBox OnEndLabelEdit
^EnableButton("Up");ChangeButtonBinding("Up", "JumpId(^gizmos.hlp',`wxeditablelistbox')")
^K OnEndLabelEdit

`$#+K!wxEditableListBox::OnItemSelected`

`void OnItemSelected(wxListEvent& event)K`

^wxEditableListBox::OnItemSelected
^wxeditablelistboxonitemselected
^browse00026
^K wxEditableListBox OnItemSelected
^EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`gizmos.hlp', `wxeditablelistbox')")
^K OnItemSelected

`wxEditableListBox::OnNewItem`

`void OnNewItem(wxCommandEvent& event)`^K

^w`xEditableListBox::OnNewItem`

^w`xeditablelistboxonnewitem`

^b`rowse00027`

^K`wxEditableListBox OnNewItem`

^E`nableButton("Up");ChangeButtonBinding("Up", "JumpId(^gizmos.hlp', `wxeditablelistbox')")`

^K`OnNewItem`

`$#+K! wxEditableListBox::OnUpItem`

`void OnUpItem(wxCommandEvent& event)K`

^wxEditableListBox::OnUpItem
^wxeditablelistboxonupitem
^browse00028
^K wxEditableListBox OnUpItem
^EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`gizmos.hlp', `wxeditablelistbox')")
^K OnUpItem

`wxEditableListBox::SetStrings`

`void SetStrings(const wxArrayString& strings)`^K

^w`xEditableListBox::SetStrings`

^w`xeditablelistboxsetstrings`

^b`rowse00029`

^K`wxEditableListBox SetStrings`

^E`nableButton("Up");ChangeButtonBinding("Up", "JumpId(`gizmos.hlp', `wxeditablelistbox')")`

^K`SetStrings`

`$#+K! wxLEDNumberCtrl::wxLEDNumberCtrl`

`wxLEDNumberCtrl(wxWindow* parent, wxWindowID id = -1, const wxPoint& pos = wxDefaultPosition, const wxSize& size = wxDefaultSize, long style = wxLED_ALIGN_LEFT | wxLED_DRAW_FADED)`^K

`wxLEDNumberCtrl()`^K

Constructors.

^w`xLEDNumberCtrl::wxLEDNumberCtrl`
^w`xlednumberctrlwxlednumberctrl`
^b`rowse00031`
^K`wxLEDNumberCtrl wxLEDNumberCtrl`
^E`nableButton("Up");ChangeButtonBinding("Up", "JumpId(`gizmos.hlp', `wxlednumberctrl')")`
^K`wxLEDNumberCtrl`
^K`wxLEDNumberCtrl`

wxLEDNumberCtrl::Create

bool Create(**wxWindow*** *parent*, **wxWindowID** *id* = -1, **const wxPoint&** *pos* = *wxDefaultPosition*, **const wxSize&** *size* = *wxDefaultSize*, **long** *style* = 0)^K

Create functions.

^wxLEDNumberCtrl::Create
^wxlednumberctrlcreate
^browse00032
^K wxLEDNumberCtrl Create
^EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`gizmos.hlp', `wxlednumberctrl')")
^K Create

`$#+KK!wxLEDNumberCtrl::GetAlignment`

`wxLEDValueAlign GetAlignment() const`

`wxLEDNumberCtrl::GetAlignment`

`wxlednumberctrlgetalignment`

`rowse00033`

`K wxLEDNumberCtrl GetAlignment`

`K GetAlignment`

`EnableButton("Up");ChangeButtonBinding("Up", "JumpId(^gizmos.hlp', `wxlednumberctrl')")`

\$#+KK!wxLEDNumberCtrl::GetDrawFaded

bool GetDrawFaded() const

wxLEDNumberCtrl::GetDrawFaded
wxlednumberctrlgetdrawfaded
rowse00034
K wxLEDNumberCtrl GetDrawFaded
K GetDrawFaded
E nableButton("Up");ChangeButtonBinding("Up", "JumpId(^gizmos.hlp',`wxlednumberctrl')")

\$#+KK!wxLEDNumberCtrl::GetValue

const wxString& GetValue() const

wxLEDNumberCtrl::GetValue
wxlednumberctrlgetvalue
browse00035
K wxLEDNumberCtrl GetValue
K GetValue
E nableButton("Up");ChangeButtonBinding("Up", "JumpId(^gizmos.hlp',`wxlednumberctrl')")

\$#+K! **wxLEDNumberCtrl::SetAlignment**

void SetAlignment(wxLEDValueAlign *Alignment*, bool *Redraw* = *TRUE*)^K

WxLEDNumberCtrl::SetAlignment

wxlednumberctrlsetalignment

browse00036

K wxLEDNumberCtrl SetAlignment

EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`gizmos.hlp', `wxlednumberctrl')")

K SetAlignment

^{\$#+K!}**wxLEDNumberCtrl::SetDrawFaded**

void SetDrawFaded(**bool** *DrawFaded*, **bool** *Redraw* = *TRUE*)^K

^wxLEDNumberCtrl::SetDrawFaded
^wxlednumberctrlsetdrawfaded
^browse00037
^K wxLEDNumberCtrl SetDrawFaded
^EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`gizmos.hlp', `wxlednumberctrl')")
^K SetDrawFaded

^{\$#+K!}**wxLEDNumberCtrl::SetValue**

void SetValue(const wxString& Value, bool Redraw = TRUE)^K

^wxLEDNumberCtrl::SetValue
^wxlednumberctrlsetvalue
^browse00038
^K wxLEDNumberCtrl SetValue
^EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`gizmos.hlp', `wxlednumberctrl')")
^K SetValue

`$#+K! wxMultiCellCanvas::wxMultiCellCanvas`

`wxMultiCellCanvas(wxWindow* parent, int numRows = 2, int numCols = 2)K`

`wxMultiCellCanvas::wxMultiCellCanvas`
`wxmulticellcanvaswxmulticellcanvas`
`browse00040`
`K wxMultiCellCanvas wxMultiCellCanvas`
`E nableButton("Up");ChangeButtonBinding("Up", "JumpId(`gizmos.hlp', `wxmulticellcanvas')")`
`K wxMultiCellCanvas`

$\$#+K!$ wxMultiCellCanvas::Add

void Add(wxWindow* *win*, unsigned int *row*, unsigned int *col*)^K

^wxMultiCellCanvas::Add
^wxmulticellcanvasadd
^browse00041
^K wxMultiCellCanvas Add
^EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`gizmos.hlp', `wxmulticellcanvas')")
^K Add

$\$#+K!$ wxMultiCellCanvas::CalculateConstraints

void CalculateConstraints()^K

^w wxMultiCellCanvas::CalculateConstraints
^w wxmulticellcanvascalculateconstraints
^b rowse00042
^K wxMultiCellCanvas CalculateConstraints
^E nableButton("Up");ChangeButtonBinding("Up", "JumpId(`gizmos.hlp', `wxmulticellcanvas')")
^K CalculateConstraints

```
$#+K!wxMultiCellCanvas::MaxCols
int MaxCols()K
```

```
wxMultiCellCanvas::MaxCols
wxmulticellcanvasmaxcols
browse00043
K wxMultiCellCanvas MaxCols
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`gizmos.hlp', `wxmulticellcanvas')")
K MaxCols
```


\$#+K!wxMultiCellCanvas::MaxRows

int MaxRows()^K

^wxMultiCellCanvas::MaxRows
^wxmulticellcanvasmaxrows
^browse00044
^K wxMultiCellCanvas MaxRows
^EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`gizmos.hlp', `wxmulticellcanvas')")
^K MaxRows

`$#+K! wxMultiCellCanvas::Resize`

`void Resize(int numRows, int numCols)K`

`w` wxMultiCellCanvas::Resize
`w` wxmulticellcanvasresize
`b` rowse00045
`K` wxMultiCellCanvas Resize
`E` nableButton("Up");ChangeButtonBinding("Up", "JumpId(`gizmos.hlp', `wxmulticellcanvas')")
`K` Resize

`$#+K! wxMultiCellCanvas::SetMinCellSize`

`void SetMinCellSize(const wxSize size)K`

`wxMultiCellCanvas::SetMinCellSize`

`wxmulticellcanvassetmincellsize`

`rowse00046`

`K wxMultiCellCanvas SetMinCellSize`

`E nableButton("Up");ChangeButtonBinding("Up", "JumpId(`gizmos.hlp', `wxmulticellcanvas')")`

`K SetMinCellSize`

`$#+K!` wxMultiCellItemHandle::wxMultiCellItemHandle

`wxMultiCellItemHandle(int row, int column, wxSize size, wxResizable style = wxNOT_RESIZABLE, wxSize weight = wxSize(1, 1), int align = wxALIGN_NOT)`^K

`wxMultiCellItemHandle(int row, int column, wxResizable style, wxSize weight = wxSize(1, 1), int align = wxALIGN_NOT)`^K

`wxMultiCellItemHandle(int row, int column, int align)`^K

`wxMultiCellItemHandle(int row, int column, int height = 1, int width = 1, wxSize size = wxDefaultSize, wxResizable style = wxNOT_RESIZABLE, wxSize weight = wxSize(1, 1), int align = wxALIGN_NOT)`^K

^w`wxMultiCellItemHandle::wxMultiCellItemHandle`

^w`wxmulticellitemhandlewxmulticellitemhandle`

^b`rowse00048`

^K`wxMultiCellItemHandle wxMultiCellItemHandle`

^E`nableButton("Up");ChangeButtonBinding("Up", "JumpId(`gizmos.hlp', `wxmulticellitemhandle')")`

^K`wxMultiCellItemHandle`

^K`wxMultiCellItemHandle`

^K`wxMultiCellItemHandle`

^K`wxMultiCellItemHandle`

$\$#+K!$ wxMultiCellItemHandle::GetAlignment

int GetAlignment()^K

^wxMultiCellItemHandle::GetAlignment
^wxmulticellitemhandlegetalignment
^browse00049
^K wxMultiCellItemHandle GetAlignment
^EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`gizmos.hlp', `wxmulticellitemhandle')")
^K GetAlignment

\$#+K!wxMultiCellItemHandle::GetColumn

int GetColumn()^K

^wxMultiCellItemHandle::GetColumn
^wxmulticellitemhandlegetcolumn
^browse00050
^K wxMultiCellItemHandle GetColumn
^EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`gizmos.hlp', `wxmulticellitemhandle')")
^K GetColumn

$\$#+K!$ wxMultiCellItemHandle::GetHeight

int GetHeight()^K

^wxMultiCellItemHandle::GetHeight
^wxmulticellitemhandlegetheight
^browse00051
^K wxMultiCellItemHandle GetHeight
^EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`gizmos.hlp', `wxmulticellitemhandle')")
^K GetHeight

\$#+K! wxMultiCellItemHandle::GetLocalSize

wxSize GetLocalSize()^K

^wxMultiCellItemHandle::GetLocalSize
^wxmulticellitemhandlegetlocalsize
^browse00052
^K wxMultiCellItemHandle GetLocalSize
^EnableButton("Up");ChangeButtonBinding("Up", "JumpId(^ gizmos.hlp', `wxmulticellitemhandle')")
^K GetLocalSize

$\$#+K!$ **wxMultiCellItemHandle::GetRow**

int GetRow()^K

^wxMultiCellItemHandle::GetRow
^wxmulticellitemhandlegetrow
^browse00053
^K wxMultiCellItemHandle GetRow
^EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`gizmos.hlp', `wxmulticellitemhandle')")
^K GetRow

$\$#+K!$ wxMultiCellItemHandle::GetStyle

wxResizable GetStyle()^K

^wxMultiCellItemHandle::GetStyle
^wxmulticellitemhandlegetstyle
^browse00054
^K wxMultiCellItemHandle GetStyle
^EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`gizmos.hlp', `wxmulticellitemhandle')")
^K GetStyle

\$#+K! wxMultiCellItemHandle::GetWeight

wxSize GetWeight()^K

^wxMultiCellItemHandle::GetWeight
^wxmulticellitemhandlegetweight
^browse00055
^K wxMultiCellItemHandle GetWeight
^EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`gizmos.hlp', `wxmulticellitemhandle')")
^K GetWeight

\$#+K!wxMultiCellItemHandle::GetWidth

int GetWidth()^K

^wxMultiCellItemHandle::GetWidth
^wxmulticellitemhandlegetwidth
^browse00056
^K wxMultiCellItemHandle GetWidth
^EnableButton("Up");ChangeButtonBinding("Up", "JumpId(^gizmos.hlp', `wxmulticellitemhandle')")
^K GetWidth

`$#+K! wxMultiCellSizer::wxMultiCellSizer`

`wxMultiCellSizer(int rows, int cols)K`

`wxMultiCellSizer(wxSize & size)K`

`w_xMultiCellSizer::wxMultiCellSizer`

`w_xmulticellsizerwxmulticellsizer`

`b_rowse00058`

`K wxMultiCellSizer wxMultiCellSizer`

`EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`gizmos.hlp', `wxmulticellsizer')")`

`K wxMultiCellSizer`

`K wxMultiCellSizer`

$\$#+K!$ wxMultiCellSizer::~wxMultiCellSizer

\sim wxMultiCellSizer()^K

w_x MultiCellSizer::~wxMultiCellSizer
 w_x multicellsizerdtor
 $b_{rowse00059}$
 $K_{wxMultiCellSizer} \sim wxMultiCellSizer$
 $E_{nableButton("Up");ChangeButtonBinding("Up", "JumpId(\`gizmos.hlp', \`wxmulticellsizer')")}$
 $K_{\sim wxMultiCellSizer}$

\$#+K! wxMultiCellSizer::CalcMin

wxSize CalcMin()^K

^wxMultiCellSizer::CalcMin
^wxmulticellsizercalcmin
^browse00060
^K wxMultiCellSizer CalcMin
^EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`gizmos.hlp', `wxmulticellsizer')")
^K CalcMin

`$#+K! wxMultiCellSizer::EnableGridLines`

`bool EnableGridLines(wxWindow* win)K`

`wxMultiCellSizer::EnableGridLines`

`wxmulticellsizerenablegridlines`

`rowse00061`

`K wxMultiCellSizer EnableGridLines`

`E nableButton("Up");ChangeButtonBinding("Up", "JumpId(^gizmos.hlp', `wxmulticellsizer')")`

`K EnableGridLines`

`$#+K! wxMultiCellSizer::OnPaint`

`void OnPaint(wxDC& dc)K`

`w_xMultiCellSizer::OnPaint`
`w_xmulticellsizeronpaint`
`browse00062`
`K wxMultiCellSizer OnPaint`
`EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`gizmos.hlp', `wxmulticellsizer')")`
`K OnPaint`

```
$#+K!wxMultiCellSizer::RecalcSizes
void RecalcSizes()K
```

```
wxMultiCellSizer::RecalcSizes
wxmulticellsizerrecalcsizes
browse00063
K wxMultiCellSizer RecalcSizes
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`gizmos.hlp', `wxmulticellsizer')")
K RecalcSizes
```

`$#+K!wxMultiCellSizer::SetColumnWidth`

`bool SetColumnWidth(int column, int colSize = 5, bool expandable = FALSE)K`

`wxMultiCellSizer::SetColumnWidth`
`wxmulticellsizersetcolumnwidth`
`browse00064`
`K wxMultiCellSizer SetColumnWidth`
`EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`gizmos.hlp', `wxmulticellsizer')")`
`K SetColumnWidth`

`$#+K! wxMultiCellSizer::SetDefaultCellSize`

`bool SetDefaultCellSize(wxSize size)K`

`w`xMultiCellSizer::SetDefaultCellSize
`w`xmulticellsizersetdefaultcellsize
`b`rowse00065
`K` wxMultiCellSizer SetDefaultCellSize
`E`nableButton("Up");ChangeButtonBinding("Up", "JumpId(`gizmos.hlp', `wxmulticellsizer')")
`K` SetDefaultCellSize

\$#+K! wxMultiCellSizer::SetGridPen

bool SetGridPen(wxPen* pen)^K

^wxMultiCellSizer::SetGridPen
^wxmulticellsizersetgridpen
^browse00066
^K wxMultiCellSizer SetGridPen
^EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`gizmos.hlp', `wxmulticellsizer')")
^K SetGridPen

`$#+K! wxMultiCellSizer::SetRowHeight`

`bool SetRowHeight(int row, int rowSize = 5, bool expandable = FALSE)K`

`wxMultiCellSizer::SetRowHeight
wxmulticellsizersetrowheight
browse00067
K wxMultiCellSizer SetRowHeight
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`gizmos.hlp', `wxmulticellsizer')")
K SetRowHeight`

`$#+K! wxRemotelyScrolledTreeCtrl::wxRemotelyScrolledTreeCtrl`

`wxRemotelyScrolledTreeCtrl(wxWindow* parent, wxWindowID id, const wxPoint& pt = wxDefaultPosition, const wxSize& sz = wxDefaultSize, long style = wxTR_HAS_BUTTONS)K`

`wxRemotelyScrolledTreeCtrl::wxRemotelyScrolledTreeCtrl
wxremotelyscrolledtreectrlwxremotelyscrolledtreectrl
browse00069
K wxRemotelyScrolledTreeCtrl wxRemotelyScrolledTreeCtrl
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`gizmos.hlp', `wxremotelyscrolledtreectrl')")
K wxRemotelyScrolledTreeCtrl`

\$#+K!wxRemotelyScrolledTreeCtrl::~wxRemotelyScrolledTreeCtrl

~wxRemotelyScrolledTreeCtrl()^K

^wxRemotelyScrolledTreeCtrl::~wxRemotelyScrolledTreeCtrl
^wxremotelyscrolledtreectrlctor
^browse00070
^K wxRemotelyScrolledTreeCtrl ~wxRemotelyScrolledTreeCtrl
^EnableButton("Up");ChangeButtonBinding("Up", "JumpId(^gizmos.hlp', `wxremotelyscrolledtreectrl')")
^K ~wxRemotelyScrolledTreeCtrl

`$#+K! wxRemotelyScrolledTreeCtrl::AdjustRemoteScrollbars`

`void AdjustRemoteScrollbars()`^K

Adjust the containing wxScrolledWindow's scrollbars appropriately

^w `wxRemotelyScrolledTreeCtrl::AdjustRemoteScrollbars`

^w `wxremotelyscrolledtreectrladjustremotescrollbars`

^b `rowse00071`

^K `wxRemotelyScrolledTreeCtrl AdjustRemoteScrollbars`

^E`nableButton("Up");ChangeButtonBinding("Up", "JumpId(^gizmos.hlp', `wxremotelyscrolledtreectrl')")`

^K `AdjustRemoteScrollbars`

wxRemotelyScrolledTreeCtrl::CalcTreeSize

void CalcTreeSize(const wxTreeItemId& *id*, wxRect& *rect*)^K

void CalcTreeSize(wxRect& *rect*)^K

Calculate the tree overall size so we can set the scrollbar correctly

^wxRemotelyScrolledTreeCtrl::CalcTreeSize

^wxremotelyscrolledtreectrlcalctreesize

^browse00072

^K wxRemotelyScrolledTreeCtrl CalcTreeSize

^EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`gizmos.hlp', `wxremotelyscrolledtreectrl')")

^K CalcTreeSize

^K CalcTreeSize

`$#+KKl wxRemotelyScrolledTreeCtrl::GetCompanionWindow`

`wxWindow* GetCompanionWindow() const`

`w_xRemotelyScrolledTreeCtrl::GetCompanionWindow`
`w_xremotelyscrolledtreectrlgetcompanionwindow`
`browse00073`
`K wxRemotelyScrolledTreeCtrl GetCompanionWindow`
`K GetCompanionWindow`
`E nableButton("Up");ChangeButtonBinding("Up", "JumpId(^gizmos.hlp', `wxremotelyscrolledtreectrl)")`

`$#+KKl wxRemotelyScrolledTreeCtrl::GetScrollPos`

`int GetScrollPos(int orient) const`

In case we're using the generic tree control.

```
w_xRemotelyScrolledTreeCtrl::GetScrollPos
w_xremotelyscrolledtreectrlgetscrollpos
browse00074
K wxRemotelyScrolledTreeCtrl GetScrollPos
K GetScrollPos
E nableButton("Up");ChangeButtonBinding("Up", "JumpId(^gizmos.hlp',`wxremotelyscrolledtreectrl')")
```

\$#+KKl wxRemotelyScrolledTreeCtrl::GetScrolledWindow

wxScrolledWindow* GetScrolledWindow() const

Find the scrolled window that contains this control

w_xRemotelyScrolledTreeCtrl::GetScrolledWindow
w_xremotelyscrolledtreectrlgetscrolledwindow
b_rowse00075
K wxRemotelyScrolledTreeCtrl GetScrolledWindow
K GetScrolledWindow
E nableButton("Up");ChangeButtonBinding("Up", "JumpId(^gizmos.hlp',`wxremotelyscrolledtreectrl)")

`$#+KKl wxRemotelyScrolledTreeCtrl::GetViewStart`

`void GetViewStart(int* x, int* y) const`

In case we're using the generic tree control. Get the view start

`w_xRemotelyScrolledTreeCtrl::GetViewStart`
`w_xremotelyscrolledtreectrlgetviewstart`
`browse00076`
`K wxRemotelyScrolledTreeCtrl GetViewStart`
`K GetViewStart`
`E nableButton("Up");ChangeButtonBinding("Up", "JumpId(^gizmos.hlp', `wxremotelyscrolledtreectrl)")`

\$#+K! wxRemotelyScrolledTreeCtrl::HideVScrollbar

void HideVScrollbar()^K

Helpers

^wxRemotelyScrolledTreeCtrl::HideVScrollbar
^wxremotelyscrolledtreectrlhidevscrollbar
^browse00077
^K wxRemotelyScrolledTreeCtrl HideVScrollbar
^EnableButton("Up");ChangeButtonBinding("Up", "JumpId(^gizmos.hlp', `wxremotelyscrolledtreectrl)")
^K HideVScrollbar

`$#+K! wxRemotelyScrolledTreeCtrl::OnExpand`

`void OnExpand(wxTreeEvent& event)K`

`w`xRemotelyScrolledTreeCtrl::OnExpand
`w`xremotelyscrolledtreectrlonexpand
`b`rowse00078
`K` wxRemotelyScrolledTreeCtrl OnExpand
`E`nableButton("Up");ChangeButtonBinding("Up", "JumpId(^gizmos.hlp', `wxremotelyscrolledtreectrl')")
`K` OnExpand

`$#+K! wxRemotelyScrolledTreeCtrl::OnPaint`

`void OnPaint(wxPaintEvent& event)K`

^w`xRemotelyScrolledTreeCtrl::OnPaint`

^w`xremotelyscrolledtreectrlonpaint`

^b`rowse00079`

^K`wxRemotelyScrolledTreeCtrl OnPaint`

^E`nableButton("Up");ChangeButtonBinding("Up", "JumpId(^gizmos.hlp', `wxremotelyscrolledtreectrl')")`

^K`OnPaint`

`$#+K! wxRemotelyScrolledTreeCtrl::OnScroll`

`void OnScroll(wxScrollWinEvent& event)K`

`w`xRemotelyScrolledTreeCtrl::OnScroll
`w`xremotelyscrolledtreectrlonscroll
`b`rowse00080
`K` wxRemotelyScrolledTreeCtrl OnScroll
`E`nableButton("Up");ChangeButtonBinding("Up", "JumpId(^gizmos.hlp', `wxremotelyscrolledtreectrl')")
`K` OnScroll

`$#+K!` **wxRemotelyScrolledTreeCtrl::OnSize**

void OnSize(wxSizeEvent& *event*)^K

Events

^wxRemotelyScrolledTreeCtrl::OnSize
^wxremotelyscrolledtreectrlonsize
^browse00081
^K wxRemotelyScrolledTreeCtrl OnSize
^EnableButton("Up");ChangeButtonBinding("Up", "JumpId(^gizmos.hlp', `wxremotelyscrolledtreectrl')")
^K OnSize

`$#+K! wxRemotelyScrolledTreeCtrl::PrepareDC`

`void PrepareDC(wxDC& dc)K`

In case we're using the generic tree control.

```
w_xRemotelyScrolledTreeCtrl::PrepareDC
w_xremotelyscrolledtreectrlpreparedc
b_rowse00082
K wxRemotelyScrolledTreeCtrl PrepareDC
E nableButton("Up");ChangeButtonBinding("Up", "JumpId(^gizmos.hlp', `wxremotelyscrolledtreectrl')")
K PrepareDC
```

`wxRemotelyScrolledTreeCtrl::ScrollToLine`

`void ScrollToLine(int posHoriz, int posVert)`^K

Scroll to the given line (in scroll units where each unit is the height of an item)

^w`wxRemotelyScrolledTreeCtrl::ScrollToLine`

^w`wxremotelyscrolledtreectrlscrolltoline`

^b`rowse00083`

^K`wxRemotelyScrolledTreeCtrl ScrollToLine`

^E`nableButton("Up");ChangeButtonBinding("Up", "JumpId(^gizmos.hlp', `wxremotelyscrolledtreectrl')")`

^K`ScrollToLine`

`$#+K! wxRemotelyScrolledTreeCtrl::SetCompanionWindow`

`void SetCompanionWindow(wxWindow* companion)`^K

Accessors The companion window is one which will get notified when certain events happen such as node expansion

^w wxRemotelyScrolledTreeCtrl::SetCompanionWindow
^w wxremotelyscrolledtreectrlsetcompanionwindow
^b rowse00084
^K wxRemotelyScrolledTreeCtrl SetCompanionWindow
^E nableButton("Up");ChangeButtonBinding("Up", "JumpId(^gizmos.hlp', `wxremotelyscrolledtreectrl)")
^K SetCompanionWindow

^{S#+K!}**wxRemotelyScrolledTreeCtrl::SetScrollbars**

void SetScrollbars(*int pixelsPerUnitX*, *int pixelsPerUnitY*, *int noUnitsX*, *int noUnitsY*,
int xPos = 0, *int yPos = 0*, **bool noRefresh = FALSE**)^K

Overrides Override this in case we're using the generic tree control. Calls to this should
disable the vertical scrollbar. Number of pixels per user unit (0 or -1 for no scrollbar)
Length of virtual canvas in user units Length of page in user units

^wwxRemotelyScrolledTreeCtrl::SetScrollbars

^wwxremotelyscrolledtreectrlsetscrollbars

^browse00085

^KwxRemotelyScrolledTreeCtrl SetScrollbars

^EnableButton("Up");ChangeButtonBinding("Up", "JumpId(^gizmos.hlp', `wxremotelyscrolledtreectrl)")

^KSetScrollbars

`$#+K! wxSplitterScrolledWindow::wxSplitterScrolledWindow`

`wxSplitterScrolledWindow(wxWindow* parent, wxWindowID id = -1, const wxPoint& pos = wxDefaultPosition, const wxSize& sz = wxDefaultSize, long style = 0)K`

`wxSplitterScrolledWindow::wxSplitterScrolledWindow`
`wxsplitterscrolledwindowwxsplitterscrolledwindow`
`browse00087`
`K wxSplitterScrolledWindow wxSplitterScrolledWindow`
`E nableButton("Up");ChangeButtonBinding("Up", "JumpId(`gizmos.hlp', `wxsplitterscrolledwindow')")`
`K wxSplitterScrolledWindow`

\$#+K! **wxSplitterScrolledWindow::OnScroll**

void OnScroll(wxScrollWinEvent& event)^K

Overrides Events

wxSplitterScrolledWindow::OnScroll

wxsplitterscrolledwindowonscroll

browse00088

K wxSplitterScrolledWindow OnScroll

EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`gizmos.hlp', `xsplitterscrolledwindow')")

K OnScroll

`wxSplitterScrolledWindow::OnSize`

`void OnSize(wxSizeEvent& event)`^K

^w`SplitterScrolledWindow::OnSize`

^w`splitterscrolledwindow::size`

^b`rowse00089`

^K`wxSplitterScrolledWindow OnSize`

^E`nableButton("Up");ChangeButtonBinding("Up", "JumpId(`gizmos.hlp', `wxsplitterscrolledwindow')")`

^K`OnSize`

`$#+K! wxThinSplitterWindow::wxThinSplitterWindow`

`wxThinSplitterWindow(wxWindow* parent, wxWindowID id = -1, const wxPoint& pos = wxDefaultPosition, const wxSize& sz = wxDefaultSize, long style = wxSP_3D | wxCLIP_CHILDREN)K`

`wxThinSplitterWindow::wxThinSplitterWindow
wxthinsplitterwindowwxthinsplitterwindow
browse00091
K wxThinSplitterWindow wxThinSplitterWindow
E nableButton("Up");ChangeButtonBinding("Up", "JumpId(^gizmos.hlp', `wxthinsplitterwindow')")
K wxThinSplitterWindow`

`$#+K! wxThinSplitterWindow::DrawSash`

`void DrawSash(wxDC& dc)K`

`wxThinSplitterWindow::DrawSash
wxthinsplitterwindowdrawsash
browse00092
K wxThinSplitterWindow DrawSash
E nableButton("Up");ChangeButtonBinding("Up", "JumpId(^ gizmos.hlp', `wxthinsplitterwindow')")
K DrawSash`

`$#+K!wxThinSplitterWindow::OnSize`

`void OnSize(wxSizeEvent& event)K`

Events

`wxThinSplitterWindow::OnSize`

`wxthinsplitterwindow::size`

`rowse00093`

`K wxThinSplitterWindow OnSize`

`EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`gizmos.hlp', `wxthinsplitterwindow')")`

`K OnSize`

`$#+K!` **wxThinSplitterWindow::SashHitTest**

bool SashHitTest(int *x*, int *y*, int *tolerance* = 2)^K

Tests for *x*, *y* over sash. Overriding this allows us to increase the tolerance.

`w`xThinSplitterWindow::SashHitTest
`w`xthinsplitterwindowsashittest
`b`rowse00094
`K` wxThinSplitterWindow SashHitTest
`E`nableButton("Up");ChangeButtonBinding("Up", "JumpId(`gizmos.hlp', `wxthinsplitterwindow')")
`K` SashHitTest

`$#+K!` **wxThinSplitterWindow::SizeWindows**

void SizeWindows()^K

Overrides

`w`xThinSplitterWindow::SizeWindows
`w`xthinsplitterwindowssizewindows
`b`rowse00095
`K` wxThinSplitterWindow SizeWindows
`E`nableButton("Up");ChangeButtonBinding("Up", "JumpId(^gizmos.hlp',`wxthinsplitterwindow')")
`K` SizeWindows

\$#+K! wxTreeCompanionWindow::wxTreeCompanionWindow

wxTreeCompanionWindow(wxWindow* parent, wxWindowID id = -1, const wxPoint& pos = wxDefaultPosition, const wxSize& sz = wxDefaultSize, long style = 0)^K

^wxTreeCompanionWindow::wxTreeCompanionWindow
^wxtreecompanionwindowwxtreecompanionwindow
^browse00097
^K wxTreeCompanionWindow wxTreeCompanionWindow
^EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`gizmos.hlp', `wtreecompanionwindow')")
^K wxTreeCompanionWindow

wxTreeCompanionWindow::DrawItem

void DrawItem(wxDC& *dc*, wxTreeItemId *id*, const wxRect& *rect*)^K

Overrides

^wxTreeCompanionWindow::DrawItem
^wxtreecompanionwindowdrawitem
^browse00098
^K wxTreeCompanionWindow DrawItem
^EnableButton("Up");ChangeButtonBinding("Up", "JumpId(^gizmos.hlp', `wtreecompanionwindow')")
^K DrawItem

\$#+KKl wxTreeCompanionWindow::GetTreeCtrl

wxRemotelyScrolledTreeCtrl* GetTreeCtrl() const

Operations Accessors

wxTreeCompanionWindow::GetTreeCtrl

wxtreecompanionwindowgettreectrl

browse00099

K wxTreeCompanionWindow GetTreeCtrl

K GetTreeCtrl

EnableButton("Up");ChangeButtonBinding("Up", "JumpId(^gizmos.hlp', `wxtreecompanionwindow')")

`$#+K! wxTreeCompanionWindow::OnExpand`

`void OnExpand(wxTreeEvent& event)K`

`w`xTreeCompanionWindow::OnExpand
`w`xtreecompanionwindowonexpand
`b`rowse00100
`K` wxTreeCompanionWindow OnExpand
`E`nableButton("Up");ChangeButtonBinding("Up", "JumpId(`gizmos.hlp', `wtreecompanionwindow')")
`K` OnExpand

`$#+K!` **wxTreeCompanionWindow::OnPaint**

void OnPaint(wxPaintEvent& *event*)^K

Events

^wxTreeCompanionWindow::OnPaint
^wxtreecompanionwindowonpaint
^browse00101
^K wxTreeCompanionWindow OnPaint
^EnableButton("Up");ChangeButtonBinding("Up", "JumpId(^gizmos.hlp', `wtreecompanionwindow')")
^K OnPaint

`$#+K! wxTreeCompanionWindow::OnScroll`

`void OnScroll(wxScrollWinEvent& event)K`

`w`xTreeCompanionWindow::OnScroll
`w`xtreecompanionwindowonscroll
`b`rowse00102
`K` wxTreeCompanionWindow OnScroll
`E`nableButton("Up");ChangeButtonBinding("Up", "JumpId(`gizmos.hlp', `wtreecompanionwindow')")
`K` OnScroll

`$#+K! wxTreeCompanionWindow::SetTreeCtrl`

`void SetTreeCtrl(wxRemotelyScrolledTreeCtrl* treeCtrl)K`

^wxTreeCompanionWindow::SetTreeCtrl
^wxtreecompanionwindowsettreectrl
^browse00103
^K wxTreeCompanionWindow SetTreeCtrl
^EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`gizmos.hlp', `wtreecompanionwindow')")
^K SetTreeCtrl

