

#\$+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](#)

^Contents

^Contents

^browse00001

^K Contents

^DisableButton("Up")

Copyright notice

The licence is the wxWindows Licence.

Copyright notice
topic0
browse00002
Copyright notice
isableButton("Up")

\$#+K! Introduction

What is the Gizmos library?

^lntroduction

ⁱntroduction

^browse00003

^K Introduction

^DisableButton("Up")

^{\$#+K!} **Alphabetical class reference**

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

^Alphabetical class reference

^Classref

^Browse00005

^K Alphabetical class reference

^DisableButton("Up")

Classes by category

A classification of Gizmos classes by category.

Classes by category
Classesbycat
rowse00104
Classes by category
isableButton("Up")

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")

What is the Gizmos library?

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

What is the Gizmos library?

topic1

browse00004

What is the Gizmos library?

EnableButton("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`

`wxdynamicsashsplitevent`

`rowse00006`

`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`

`wxdynamicsashunifyevent`

`rowse00009`

`wxDynamicSashUnifyEvent`

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

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

wxDynamicSashWindow. See above.

Derived from

wxWindow

Data structures

Members

wxDynamicSashWindow::wxDynamicSashWindow
wxDynamicSashWindow::~~wxDynamicSashWindow
wxDynamicSashWindow::AddChild
wxDynamicSashWindow::Create
wxDynamicSashWindow::GetHScrollBar
wxDynamicSashWindow::GetVScrollBar

^wxDynamicSashWindow

^wxdynamicsashwindow

^browse00012

^K wxDynamicSashWindow

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

`##+KL` **wxEitableListBox**

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

Derived from

wxPanel

Data structures

Members

wxEitableListBox::wxEitableListBox
wxEitableListBox::GetStrings
wxEitableListBox::OnDelItem
wxEitableListBox::OnDownItem
wxEitableListBox::OnEditItem
wxEitableListBox::OnEndLabelEdit
wxEitableListBox::OnItemSelected
wxEitableListBox::OnNewItem
wxEitableListBox::OnUpItem
wxEitableListBox::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](#)

`wxMultiCellCanvas`

`wxmultiplicellcanvas`

`rowse00039`

`K wxMultiCellCanvas`

`E enableButton("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')")

`##+KL` **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](#)

`wxRemotelyScrolledTreeCtrl`

`wxremotelyscrolledtreectrl`

`rowse00068`

`wxRemotelyScrolledTreeCtrl`

`enableButton("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')")

`$#+K!` **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

^wxthinsplitterwindow

^browse00090

^K wxThinSplitterWindow

^EnableButton("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

^K Notes on using the reference

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

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

wxDynamicSashSplitEvent(const wxDynamicSashSplitEvent& *event*)^K

wxDynamicSashSplitEvent(wxObject* *target*)^K

wxDynamicSashSplitEvent()^K

^wxDynamicSashSplitEvent::wxDynamicSashSplitEvent

^wxdynamicsashspliteventwxdynamicsshplitevent

^browse00007

^K wxDynamicSashSplitEvent wxDynamicSashSplitEvent

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

`wxdynamicsshplitevent')")

^K wxDynamicSashSplitEvent

^K wxDynamicSashSplitEvent

^K wxDynamicSashSplitEvent

`$#+KK!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

^wxDynamicSashUnifyEvent::wxDynamicSashUnifyEvent
^wxdynamicsashunifyeventwxdynamicsshunifyevent
^browse00010
^K wxDynamicSashUnifyEvent wxDynamicSashUnifyEvent
^EnableButton("Up");ChangeButtonBinding("Up", "JumpId(' gizmos.hlp',
`wxdynamicsshunifyevent')")
^K wxDynamicSashUnifyEvent
^K wxDynamicSashUnifyEvent
^K wxDynamicSashUnifyEvent

`$#+KK!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

^wxdynamicsashwindowwxdynamicssashwindow

^browse00013

^K wxDynamicSashWindow wxDynamicSashWindow

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

[`]xdynamicsashwindow')")

^K wxDynamicSashWindow

^K wxDynamicSashWindow

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

~wxDynamicSashWindow()^K

^wxDynamicSashWindow::~~wxDynamicSashWindow

^wxdynamicsashwindowdtor

^browse00014

^K wxDynamicSashWindow ~wxDynamicSashWindow

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

^K ~wxDynamicSashWindow

`wxDynamicSashWindow::AddChild`

`void AddChild(wxWindowBase* child)`

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

`wxDynamicSashWindow::AddChild`

`wxdynamicsashwindowaddchild`

`rowse00015`

`wxDynamicSashWindow AddChild`

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

`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

^wxdynamicsshwindowcreate

^browse00016

^K wxDynamicSashWindow Create

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

^K Create

`$#+KK!wxDynamicSashWindow::GetHScrollBar`

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

`wxDynamicSashWindow::GetHScrollBar`

`wxdynamicsashwindowgethscrollbar`

`browse00017`

`K wxDynamicSashWindow GetHScrollBar`

`K GetHScrollBar`

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

`$#+KK!wxDynamicSashWindow::GetVScrollBar`

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

`wxDynamicSashWindow::GetVScrollBar`

`wxdynamicsashwindowgetvscrollbar`

`browse00018`

`K wxDynamicSashWindow GetVScrollBar`

`K GetVScrollBar`

`EnableButton("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`wxEditableListBox::wxEditableListBox`

^w`wxeditablelistboxwxeditablelistbox`

^b`rowse00020`

^K `wxEditableListBox wxEditableListBox`

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

``wxeditablelistbox')")`

^K `wxEditableListBox`

`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`

`wxEditableListBox::OnDownItem`

`void OnDownItem(wxCommandEvent& event)`^K

`wxEditableListBox::OnDownItem`

`wxeditablelistboxondownitem`

`rowse00023`

`wxEditableListBox OnDownItem`

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

`OnDownItem`

`wxEditableListBox::OnEditItem`

`void OnEditItem(wxCommandEvent& event)`^K

^w`xEditableListBox::OnEditItem`

^w`xeditablelistboxonedititem`

^b`rowse00024`

^K `wxEditableListBox OnEditItem`

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

^K `OnEditItem`

\$#+K! **wxEitableListBox::OnEndLabelEdit**

void OnEndLabelEdit(wxListEvent& *event*)^K

wxEditableListBox::OnEndLabelEdit

wxeditablelistboxonendlabeledit

browse00025

K wxEditableListBox OnEndLabelEdit

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

`wxeditablelistbox')

K OnEndLabelEdit

\$#+K! **wxEitableListBox::OnItemSelected**

void OnItemSelected(wxListEvent& *event*)^K

wxEditableListBox::OnItemSelected

wxeditablelistboxonitemselected

browse00026

K wxEditableListBox OnItemSelected

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

`wxeditablelistbox')

K OnItemSelected

\$#+K! **wxEitableListBox::OnNewItem**

void OnNewItem(wxCommandEvent& *event*)^K

wxEditableListBox::OnNewItem

wxeditablelistboxonnewitem

browse00027

K wxEditableListBox OnNewItem

EnableButton("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`

\$#+K! **wxEitableListBox::SetStrings**

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

wxEditableListBox::SetStrings

wxeditablelistboxsetstrings

browse00029

K wxEditableListBox SetStrings

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

`wxeditablelistbox')

K SetStrings

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.

^wxLEDNumberCtrl::wxLEDNumberCtrl

^wxlednumberctrlwxlednumberctrl

^browse00031

^K wxLEDNumberCtrl wxLEDNumberCtrl

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

^K wxLEDNumberCtrl

^K wxLEDNumberCtrl

`$#+K! 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`

`browse00034`

`K wxLEDNumberCtrl GetDrawFaded`

`K GetDrawFaded`

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

`$#+KK!wxLEDNumberCtrl::GetValue`

`const wxString& GetValue() const`

`wxLEDNumberCtrl::GetValue`

`wxlednumberctrlgetvalue`

`rowse00035`

`K wxLEDNumberCtrl GetValue`

`K GetValue`

`EnableButton("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

^w**wxMultiCellCanvas::wxMultiCellCanvas**

^w**wxmulticellcanvaswxmulticellcanvas**

^b**rowse00040**

^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

^wxMultiCellCanvas::CalculateConstraints

^wxmulticellcanvascalculateconstraints

^browse00042

^K wxMultiCellCanvas CalculateConstraints

^EnableButton("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`xMultiCellCanvas::Resize`

^w`xmulticellcanvasresize`

^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`

`browse00046`

`K wxMultiCellCanvas SetMinCellSize`

`EnableButton("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

^browse00048

^K wxMultiCellItemHandle wxMultiCellItemHandle

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

[`]wxmulticellitemhandle")")

^K wxMultiCellItemHandle

^K wxMultiCellItemHandle

^K wxMultiCellItemHandle

^K wxMultiCellItemHandle

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

int GetAlignment()^K

w_x MultiCellItemHandle::GetAlignment

w_x multicellitemhandlegetalignment

$b_{rowse00049}$

K wxMultiCellItemHandle GetAlignment

**$E_{nableButton("Up");ChangeButtonBinding("Up", "JumpId(^gizmos.hlp',$
 $\backslash wxmulticellitemhandle')"$**

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`

`wxMultiCellSizer::wxMultiCellSizer`

`wxmulticellsizerwxmulticellsizer`

`browse00058`

`K wxMultiCellSizer wxMultiCellSizer`

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

`K wxMultiCellSizer`

`K wxMultiCellSizer`

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

~wxMultiCellSizer()^K

wxMultiCellSizer::~wxMultiCellSizer

wxmulticellsizerdtor

browse00059

K wxMultiCellSizer ~wxMultiCellSizer

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

K ~wxMultiCellSizer

\$#+K! wxMultiCellSizer::CalcMin

wxSize CalcMin()^K

^wxMultiCellSizer::CalcMin

^wxmulticellsizercalcmin

^browse00060

^K wxMultiCellSizer CalcMin

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

[`]wxmultiplicellsizer")

^K CalcMin

`$#+K! wxMultiCellSizer::EnableGridLines`

`bool EnableGridLines(wxWindow* win)K`

`wxMultiCellSizer::EnableGridLines`

`wxmulticellsizerenablegridlines`

`browse00061`

`K wxMultiCellSizer EnableGridLines`

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

`K EnableGridLines`

\$#+K! wxMultiCellSizer::OnPaint

void OnPaint(wxDC& dc)^K

^wxMultiCellSizer::OnPaint

^wxmulticellsizeronpaint

^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`

`wxMultiCellSizer::SetDefaultCellSize`

`wxmulticellsizersetdefaultcellsize`

`browse00065`

`K wxMultiCellSizer SetDefaultCellSize`

`EnableButton("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

`wxRemotelyScrolledTreeCtrl::AdjustRemoteScrollbars`

`void AdjustRemoteScrollbars()`^K

Adjust the containing wxScrolledWindow's scrollbars appropriately

^w`xRemotelyScrolledTreeCtrl::AdjustRemoteScrollbars`

^w`xremotelyscrolledtreectrladjustremotescrollbars`

^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

`$#+KK!wxRemotelyScrolledTreeCtrl::GetCompanionWindow`

`wxWindow* GetCompanionWindow() const`

`wxRemotelyScrolledTreeCtrl::GetCompanionWindow`

`wxremotelyscrolledtreectrlgetcompanionwindow`

`rowse00073`

`K wxRemotelyScrolledTreeCtrl GetCompanionWindow`

`K GetCompanionWindow`

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

`$#+KK!` **wxRemotelyScrolledTreeCtrl::GetScrollPos**

int GetScrollPos(int *orient*) const

In case we're using the generic tree control.

```
wxRemotelyScrolledTreeCtrl::GetScrollPos
wxremotelyscrolledtreectrlgetscrollpos
browse00074
K wxRemotelyScrolledTreeCtrl GetScrollPos
K GetScrollPos
E enableButton("Up");ChangeButtonBinding("Up", "JumpId(`gizmos.hlp',
`wxremotelyscrolledtreectrl')")
```

`$#+KK!` **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')")`

`$#+KK!` **wxRemotelyScrolledTreeCtrl::GetViewStart**

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

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

`wxRemotelyScrolledTreeCtrl::GetViewStart`
`wxremotelyscrolledtreectrlgetviewstart`
`rowse00076`
`K wxRemotelyScrolledTreeCtrl GetViewStart`
`K GetViewStart`
`EnableButton("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`

`wxRemotelyScrolledTreeCtrl::OnExpand`

`wxremotelyscrolledtreectrlonexpand`

`browse00078`

`K wxRemotelyScrolledTreeCtrl OnExpand`

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

``wxremotelyscrolledtreectrl')")`

`K OnExpand`

`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`

`wxRemotelyScrolledTreeCtrl::OnScroll`

`wxremotelyscrolledtreectrlonscroll`

`browse00080`

`K wxRemotelyScrolledTreeCtrl OnScroll`

`EnableButton("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.

`wxRemotelyScrolledTreeCtrl::PrepareDC`

`wxremotelyscrolledtreectrlpreparedc`

`browse00082`

`K wxRemotelyScrolledTreeCtrl PrepareDC`

`EnableButton("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)

^wxRemotelyScrolledTreeCtrl::ScrollToLine

^wxremotelyscrolledtreectrlscrolltoline

^browse00083

^K wxRemotelyScrolledTreeCtrl ScrollToLine

^EnableButton("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

^wxRemotelyScrolledTreeCtrl::SetCompanionWindow

^wxremotelyscrolledtreectrlsetcompanionwindow

^browse00084

^K wxRemotelyScrolledTreeCtrl SetCompanionWindow

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

^K SetCompanionWindow

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

^wxRemotelyScrolledTreeCtrl::SetScrollbars

^wxremotelyscrolledtreectrlsetscrollbars

^browse00085

^K wxRemotelyScrolledTreeCtrl SetScrollbars

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

^K SetScrollbars

^{\$#+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
^EnableButton("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',
`wxsplitterscrolledwindow')")

^K OnScroll

`wxSplitterScrolledWindow::OnSize`

`void OnSize(wxSizeEvent& event)`^K

^w`wxSplitterScrolledWindow::OnSize`

^w`wxsplitterscrolledwindow::OnSize`

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

wxThinSplitterWindow::DrawSash

void DrawSash(wxDC& dc)

wxThinSplitterWindow::DrawSash

wxthinsplitterwindowdrawsash

rowse00092

wxThinSplitterWindow DrawSash

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

DrawSash

`$#+K! wxThinSplitterWindow::OnSize`

`void OnSize(wxSizeEvent& event)K`

Events

`wxThinSplitterWindow::OnSize`

`wxthinsplitterwindow::OnSize`

`browse00093`

`K wxThinSplitterWindow OnSize`

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

``wxthinsplitterwindow')")`

`K OnSize`

`wxThinSplitterWindow::SashHitTest`

`bool SashHitTest(int x, int y, int tolerance = 2)`

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

`wxThinSplitterWindow::SashHitTest`

`wxthinsplitterwindowsashittest`

`rowse00094`

`wxThinSplitterWindow SashHitTest`

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

`SashHitTest`

`$#+K!wxThinSplitterWindow::SizeWindows`

`void SizeWindows()`^K

Overrides

`wxThinSplitterWindow::SizeWindows`

`wxthinsplitterwindowssizewindows`

`browse00095`

`K wxThinSplitterWindow SizeWindows`

`EnableButton("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',
`wxtreecompanionwindow')")
^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',
`wxtreecompanionwindow')")

^K DrawItem

`$#+KK!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`

^wxTreeCompanionWindow::OnExpand

^wxtreecompanionwindowonexpand

^browse00100

^K wxTreeCompanionWindow OnExpand

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

^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',
`wxtreecompanionwindow')")

^K OnPaint

^{\$#+K!}**wxTreeCompanionWindow::OnScroll**

void OnScroll(wxScrollWinEvent& *event*)^K

^wxTreeCompanionWindow::OnScroll

^wxtreecompanionwindowonscroll

^browse00102

^K wxTreeCompanionWindow OnScroll

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

^K OnScroll

`$#+K! wxTreeCompanionWindow::SetTreeCtrl`

`void SetTreeCtrl(wxRemotelyScrolledTreeCtrl* treeCtrl)K`

^w`xTreeCompanionWindow::SetTreeCtrl`

^w`xtreecompanionwindowsettreectrl`

^b`rowse00103`

^K `wxTreeCompanionWindow SetTreeCtrl`

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

^K `SetTreeCtrl`

