

#\$+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](#)

---

<sup>C</sup>ontents

<sup>C</sup>ontents

<sup>b</sup>rowse00001

<sup>K</sup> Contents

<sup>D</sup>isableButton("Up")

**Copyright notice**

The licence is the wxWindows Licence.

---

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

## **Introduction**

What is the Gizmos library?

---

Introduction  
Introduction  
Browse00003  
Introduction  
DisableButton("Up")

<sup>\$#+K!</sup> **Alphabetical class reference**

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

---

<sup>A</sup>lphabetical class reference

<sup>C</sup>lassref

<sup>B</sup>rowse00005

<sup>K</sup> Alphabetical class reference

<sup>D</sup>isableButton("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')")`

<sup>\$#+K!</sup>**wxDynamicSashWindow**

wxDynamicSashWindow. See above.

### Derived from

wxWindow

### Data structures

### Members

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

---

<sup>w</sup>xDynamicSashWindow

<sup>w</sup>xdynamicsashwindow

<sup>b</sup>rowse00012

<sup>K</sup> wxDynamicSashWindow

<sup>E</sup>nableButton("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

---

<sup>w</sup>xEditableListBox

<sup>w</sup>xeditablelistbox

<sup>b</sup>rowse00019

<sup>K</sup> wxEditableListBox

<sup>E</sup>nableButton("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

---

<sup>w</sup>xLEDNumberCtrl

<sup>w</sup>xlednumberctrl

<sup>b</sup>rowse00030

<sup>K</sup> wxLEDNumberCtrl

<sup>E</sup>nableButton("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

---

<sup>w</sup>xMultiCellItemHandle

<sup>w</sup>xmulticellitemhandle

<sup>b</sup>rowse00047

<sup>K</sup> wxMultiCellItemHandle

<sup>E</sup>nableButton("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

---

<sup>w</sup>xMultiCellSizer

<sup>w</sup>xmulticellsizer

<sup>b</sup>rowse00057

<sup>K</sup> wxMultiCellSizer

<sup>E</sup>nableButton("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

---

<sup>w</sup>xSplitterScrolledWindow

<sup>w</sup>xsplitterscrolledwindow

<sup>b</sup>rowse00086

<sup>K</sup> wxSplitterScrolledWindow

<sup>E</sup>nableButton("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

---

<sup>w</sup>xThinSplitterWindow

<sup>w</sup>xthinsplitterwindow

<sup>b</sup>rowse00090

<sup>K</sup> wxThinSplitterWindow

<sup>E</sup>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

---

<sup>w</sup>xTreeCompanionWindow

<sup>w</sup>xtreecompanionwindow

<sup>b</sup>rowse00096

<sup>K</sup> wxTreeCompanionWindow

<sup>E</sup>nableButton("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.

---

<sup>N</sup>otes on using the reference

<sup>r</sup>eferencenotes

<sup>b</sup>rowse00106

<sup>K</sup> Notes on using the reference

<sup>E</sup>nableButton("Up");ChangeButtonBinding("Up", "JumpId('gizmos.hlp', `overviews')")







<sup>\$#+K!</sup>**wxDynamicSashSplitEvent::wxDynamicSashSplitEvent**

**wxDynamicSashSplitEvent(const wxDynamicSashSplitEvent& *event*)**<sup>K</sup>

**wxDynamicSashSplitEvent(wxObject\* *target*)**<sup>K</sup>

**wxDynamicSashSplitEvent()**<sup>K</sup>

---

<sup>w</sup>xDynamicSashSplitEvent::wxDynamicSashSplitEvent

<sup>w</sup>xdynamicsashspliteventwxdynamicsshplitevent

<sup>b</sup>rowse00007

<sup>K</sup> wxDynamicSashSplitEvent wxDynamicSashSplitEvent

<sup>E</sup>nableButton("Up");ChangeButtonBinding("Up", "JumpId( gizmos.hlp',

<sup>`</sup>wxdynamicsashsplitevent')

<sup>K</sup> wxDynamicSashSplitEvent

<sup>K</sup> wxDynamicSashSplitEvent

<sup>K</sup> wxDynamicSashSplitEvent

`$#+KK!wxDynamicSashSplitEvent::Clone`

`wxEvent* Clone() const`

---

`wxDynamicSashSplitEvent::Clone`

`wxdynamicsashspliteventclone`

`browse00008`

`K wxDynamicSashSplitEvent Clone`

`K Clone`

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

<sup>\$#+K!</sup>**wxDynamicSashUnifyEvent::wxDynamicSashUnifyEvent**  
**wxDynamicSashUnifyEvent(const wxDynamicSashUnifyEvent& *event*)**<sup>K</sup>  
**wxDynamicSashUnifyEvent(wxObject\* *target*)**<sup>K</sup>  
**wxDynamicSashUnifyEvent()**<sup>K</sup>

---

<sup>w</sup>xDynamicSashUnifyEvent::wxDynamicSashUnifyEvent  
<sup>w</sup>xdynamicsashunifyeventwxdynamicssashunifyevent  
<sup>b</sup>rowse00010  
<sup>K</sup> wxDynamicSashUnifyEvent wxDynamicSashUnifyEvent  
<sup>E</sup>nableButton("Up");ChangeButtonBinding("Up", "JumpId(' gizmos.hlp',  
`wxdynamicssashunifyevent')")  
<sup>K</sup> wxDynamicSashUnifyEvent  
<sup>K</sup> wxDynamicSashUnifyEvent  
<sup>K</sup> wxDynamicSashUnifyEvent

`$#+KK!wxDynamicSashUnifyEvent::Clone`

`wxEvent* Clone() const`

---

`wxDynamicSashUnifyEvent::Clone`  
`wxdynamicsashunifyeventclone`  
`browse00011`  
`K wxDynamicSashUnifyEvent Clone`  
`K Clone`

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

<sup>\$#+K!</sup>**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"*)<sup>K</sup>

**wxDynamicSashWindow()**<sup>K</sup>

---

<sup>w</sup>xDynamicSashWindow::wxDynamicSashWindow

<sup>w</sup>xdynamicsashwindowwxdynamicssashwindow

<sup>b</sup>rowse00013

<sup>K</sup> wxDynamicSashWindow wxDynamicSashWindow

<sup>E</sup>nableButton("Up");ChangeButtonBinding("Up", "JumpId( gizmos.hlp',

<sup>`</sup>xdynamicsashwindow')")

<sup>K</sup> wxDynamicSashWindow

<sup>K</sup> wxDynamicSashWindow

<sup>\$#+K!</sup>**wxDynamicSashWindow::~~wxDynamicSashWindow**

**~wxDynamicSashWindow()**<sup>K</sup>

---

<sup>w</sup>xDynamicSashWindow::~~wxDynamicSashWindow

<sup>w</sup>xdynamicsashwindowdtor

<sup>b</sup>rowse00014

<sup>K</sup> wxDynamicSashWindow ~wxDynamicSashWindow

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

<sup>K</sup> ~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`

`wxdynamicsashwindowcreate`

`browse00016`

`K wxDynamicSashWindow Create`

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

`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`**

---

<sup>w</sup>`wxEditableListBox::wxEditableListBox`

<sup>w</sup>`weditablelistboxwxeditablelistbox`

<sup>b</sup>`rowse00020`

<sup>K</sup> `wxEditableListBox wxEditableListBox`

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

``weditablelistbox')")`

<sup>K</sup> `wxEditableListBox`

**`wxEditableListBox::GetStrings`**

**`void GetStrings(wxArrayString& strings)`**<sup>K</sup>

---

<sup>w</sup>`xEditableListBox::GetStrings`

<sup>w</sup>`xeditablelistboxgetstrings`

<sup>b</sup>`rowse00021`

<sup>K</sup> `wxEditableListBox GetStrings`

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

<sup>K</sup> `GetStrings`

<sup>\$#+K!</sup>**wxEitableListBox::OnDelItem**

**void OnDelItem(wxCommandEvent& *event*)**<sup>K</sup>

---

<sup>w</sup>xEitableListBox::OnDelItem

<sup>w</sup>xeditablelistboxondelitem

<sup>b</sup>rowse00022

<sup>K</sup> wxEitableListBox OnDelItem

<sup>E</sup>nableButton("Up");ChangeButtonBinding("Up", "JumpId(^ gizmos.hlp',

<sup>`</sup>wxeditablelistbox')

<sup>K</sup> OnDelItem

**`wxEditableListBox::OnDownItem`**

**`void OnDownItem(wxCommandEvent& event)`**<sup>K</sup>

---

`wxEditableListBox::OnDownItem`

`wxeditablelistboxondownitem`

`rowse00023`

`wxEditableListBox OnDownItem`

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

`OnDownItem`

**`wxEditableListBox::OnEditItem`**

**`void OnEditItem(wxCommandEvent& event)`**<sup>K</sup>

---

<sup>w</sup>xEditableListBox::OnEditItem

<sup>w</sup>xeditablelistboxonedititem

<sup>b</sup>rowse00024

<sup>K</sup> wxEditableListBox OnEditItem

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

<sup>K</sup> OnEditItem

**`wxEditableListBox::OnEndLabelEdit`**

**`void OnEndLabelEdit(wxListEvent& event)`**<sup>K</sup>

---

<sup>w</sup>`xEditableListBox::OnEndLabelEdit`

<sup>w</sup>`xeditablelistboxonendlabeledit`

<sup>b</sup>`rowse00025`

<sup>K</sup> `wxEditableListBox OnEndLabelEdit`

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

``wxeditablelistbox')")`

<sup>K</sup> `OnEndLabelEdit`



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

**void OnItemSelected(wxListEvent& *event*)**<sup>K</sup>

---

wxEditableListBox::OnItemSelected

wxeditablelistboxonitemselected

browse00026

K wxEditableListBox OnItemSelected

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

`wxeditablelistbox')

K OnItemSelected

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

**void OnNewItem(wxCommandEvent& *event*)**<sup>K</sup>

---

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*)**<sup>K</sup>

---

wxEditableListBox::SetStrings

wxeditablelistboxsetstrings

browse00029

K wxEditableListBox SetStrings

EnableButton("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.

---

**`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')")`

<sup>\$#+K!</sup>**wxLEDNumberCtrl::SetAlignment**

**void SetAlignment**(wxLEDValueAlign *Alignment*, **bool** *Redraw* = *TRUE*)<sup>K</sup>

---

<sup>w</sup>xLEDNumberCtrl::SetAlignment

<sup>w</sup>xlednumberctrlsetalignment

<sup>b</sup>rowse00036

<sup>K</sup> wxLEDNumberCtrl SetAlignment

<sup>E</sup>nableButton("Up");ChangeButtonBinding("Up", "JumpId(^gizmos.hlp',  
`wxlednumberctrl')")

<sup>K</sup> SetAlignment

<sup>\$#+K!</sup>**wxLEDNumberCtrl::SetDrawFaded**

**void SetDrawFaded**(**bool** *DrawFaded*, **bool** *Redraw = TRUE*)<sup>K</sup>

---

<sup>w</sup>xLEDNumberCtrl::SetDrawFaded

<sup>w</sup>xlednumberctrlsetdrawfaded

<sup>b</sup>rowse00037

<sup>K</sup> wxLEDNumberCtrl SetDrawFaded

<sup>E</sup>nableButton("Up");ChangeButtonBinding("Up", "JumpId(^gizmos.hlp',  
`wxlednumberctrl')")

<sup>K</sup> 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`

<sup>\$#+K!</sup>**wxMultiCellCanvas::wxMultiCellCanvas**

**wxMultiCellCanvas**(wxWindow\* *parent*, **int** *numRows* = 2, **int** *numCols* = 2)<sup>K</sup>

---

<sup>w</sup>xMultiCellCanvas::wxMultiCellCanvas

<sup>w</sup>xmulticellcanvaswxmulticellcanvas

<sup>b</sup>rowse00040

<sup>K</sup> wxMultiCellCanvas wxMultiCellCanvas

<sup>E</sup>nableButton("Up");ChangeButtonBinding("Up", "JumpId(^gizmos.hlp',

<sup>`</sup>wxmulticellcanvas')

<sup>K</sup> wxMultiCellCanvas

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

**void Add(wxWindow\* *win*, unsigned int *row*, unsigned int *col*)<sup>K</sup>**

---

<sup>w</sup>xMultiCellCanvas::Add

<sup>w</sup>xmulticellcanvasadd

<sup>b</sup>rowse00041

<sup>K</sup> wxMultiCellCanvas Add

<sup>E</sup>nableButton("Up");ChangeButtonBinding("Up", "JumpId(^gizmos.hlp',  
`wxmulticellcanvas')")

<sup>K</sup> Add

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

**void CalculateConstraints()**<sup>K</sup>

---

<sup>w</sup>xMultiCellCanvas::CalculateConstraints

<sup>w</sup>xmulticellcanvascalculateconstraints

<sup>b</sup>rowse00042

<sup>K</sup> wxMultiCellCanvas CalculateConstraints

<sup>E</sup>nableButton("Up");ChangeButtonBinding("Up", "JumpId(^ gizmos.hlp',  
`wxmulticellcanvas')")

<sup>K</sup> CalculateConstraints

\$#+K! **wxMultiCellCanvas::MaxCols**

**int MaxCols()**<sup>K</sup>

---

wxMultiCellCanvas::MaxCols

wxmulticellcanvasmaxcols

browse00043

K wxMultiCellCanvas MaxCols

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

K MaxCols



<sup>\$#+K!</sup>**wxMultiCellCanvas::MaxRows**

**int MaxRows()**<sup>K</sup>

---

<sup>w</sup>xMultiCellCanvas::MaxRows

<sup>w</sup>xmulticellcanvasmaxrows

<sup>b</sup>rowse00044

<sup>K</sup> wxMultiCellCanvas MaxRows

<sup>E</sup>nableButton("Up");ChangeButtonBinding("Up", "JumpId(^ gizmos.hlp',  
`wxmulticellcanvas')")

<sup>K</sup> MaxRows

**`$#+K! wxMultiCellCanvas::Resize`**

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

---

<sup>w</sup>`xMultiCellCanvas::Resize`

<sup>w</sup>`xmulticellcanvasresize`

<sup>b</sup>`rowse00045`

<sup>K</sup> `wxMultiCellCanvas Resize`

<sup>E</sup>`nableButton("Up");ChangeButtonBinding("Up", "JumpId(^ gizmos.hlp',  
`wxmulticellcanvas')")`

<sup>K</sup> `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`**

---

<sup>w</sup>`wxMultiCellItemHandle::wxMultiCellItemHandle`

<sup>w</sup>`wxmulticellitemhandlewxmulticellitemhandle`

<sup>b</sup>`rowse00048`

<sup>K</sup>`wxMultiCellItemHandle wxMultiCellItemHandle`

<sup>E</sup>`nableButton("Up");ChangeButtonBinding("Up", "JumpId(\`gizmos.hlp',  
`wxmulticellitemhandle')")`

<sup>K</sup>`wxMultiCellItemHandle`

<sup>K</sup>`wxMultiCellItemHandle`

<sup>K</sup>`wxMultiCellItemHandle`

<sup>K</sup>`wxMultiCellItemHandle`

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

**int GetAlignment()<sup>K</sup>**

---

<sup>w</sup>xMultiCellItemHandle::GetAlignment

<sup>w</sup>xmulticellitemhandlegetalignment

<sup>b</sup>rowse00049

<sup>K</sup> wxMultiCellItemHandle GetAlignment

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

<sup>K</sup> GetAlignment

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

**int GetColumn()**<sup>K</sup>

---

wxMultiCellItemHandle::GetColumn

wxmulticellitemhandlegetcolumn

browse00050

K wxMultiCellItemHandle GetColumn

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

K GetColumn

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

**int GetHeight()**<sup>K</sup>

---

<sup>w</sup>xMultiCellItemHandle::GetHeight

<sup>w</sup>xmulticellitemhandlegetheight

<sup>b</sup>rowse00051

<sup>K</sup> wxMultiCellItemHandle GetHeight

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

<sup>K</sup> GetHeight

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

**wxSize GetLocalSize()**<sup>K</sup>

---

<sup>w</sup>xMultiCellItemHandle::GetLocalSize

<sup>w</sup>xmulticellitemhandlegetlocalsize

<sup>b</sup>rowse00052

<sup>K</sup> wxMultiCellItemHandle GetLocalSize

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

<sup>K</sup> GetLocalSize



`$#+K! wxMultiCellItemHandle::GetRow`

`int GetRow()`<sup>K</sup>

---

`wxMultiCellItemHandle::GetRow`

`wxmulticellitemhandlegetrow`

`browse00053`

`K wxMultiCellItemHandle GetRow`

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

`K GetRow`

<sup>\$#+K!</sup>**wxMultiCellItemHandle::GetStyle**

**wxResizable GetStyle()**<sup>K</sup>

---

<sup>w</sup>xMultiCellItemHandle::GetStyle

<sup>w</sup>xmulticellitemhandlegetstyle

<sup>b</sup>rowse00054

<sup>K</sup> wxMultiCellItemHandle GetStyle

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

<sup>K</sup> GetStyle

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

**wxSize GetWeight()**<sup>K</sup>

---

<sup>w</sup>xMultiCellItemHandle::GetWeight

<sup>w</sup>xmulticellitemhandlegetweight

<sup>b</sup>rowse00055

<sup>K</sup> wxMultiCellItemHandle GetWeight

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

<sup>K</sup> GetWeight

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

**int GetWidth()**<sup>K</sup>

---

<sup>w</sup>xMultiCellItemHandle::GetWidth

<sup>w</sup>xmulticellitemhandlegetwidth

<sup>b</sup>rowse00056

<sup>K</sup> wxMultiCellItemHandle GetWidth

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

<sup>K</sup> 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()**<sup>K</sup>

---

wxMultiCellSizer::~wxMultiCellSizer

wxmulticellsizerdtor

browse00059

K wxMultiCellSizer ~wxMultiCellSizer

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

K ~wxMultiCellSizer

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

**wxSize CalcMin()**<sup>K</sup>

---

<sup>w</sup>xMultiCellSizer::CalcMin

<sup>w</sup>xmulticellsizercalcmin

<sup>b</sup>rowse00060

<sup>K</sup> wxMultiCellSizer CalcMin

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

<sup>K</sup> 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()`**<sup>K</sup>

---

`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`**

<sup>\$#+K!</sup>**wxRemotelyScrolledTreeCtrl::wxRemotelyScrolledTreeCtrl**

**wxRemotelyScrolledTreeCtrl**(**wxWindow\*** *parent*, **wxWindowID** *id*, **const wxPoint&** *pt* = *wxDefaultPosition*, **const wxSize&** *sz* = *wxDefaultSize*, **long** *style* = *wxTR\_HAS\_BUTTONS*)<sup>K</sup>

---

<sup>w</sup>xRemotelyScrolledTreeCtrl::wxRemotelyScrolledTreeCtrl  
<sup>w</sup>xremotelyscrolledtreectrlwxremotelyscrolledtreectrl  
<sup>b</sup>rowse00069  
<sup>K</sup> wxRemotelyScrolledTreeCtrl wxRemotelyScrolledTreeCtrl  
<sup>E</sup>nableButton("Up");ChangeButtonBinding("Up", "JumpId(^gizmos.hlp',  
`wxremotelyscrolledtreectrl')")  
<sup>K</sup> wxRemotelyScrolledTreeCtrl

<sup>\$#+K!</sup>**wxRemotelyScrolledTreeCtrl::~wxRemotelyScrolledTreeCtrl**

**~wxRemotelyScrolledTreeCtrl()**<sup>K</sup>

---

<sup>w</sup>xRemotelyScrolledTreeCtrl::~wxRemotelyScrolledTreeCtrl

<sup>w</sup>xremotelyscrolledtreectrlctor

<sup>b</sup>rowse00070

<sup>K</sup> wxRemotelyScrolledTreeCtrl ~wxRemotelyScrolledTreeCtrl

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

<sup>K</sup> ~wxRemotelyScrolledTreeCtrl



**`wxRemotelyScrolledTreeCtrl::AdjustRemoteScrollbars`**

**`void AdjustRemoteScrollbars()`**<sup>K</sup>

Adjust the containing wxScrolledWindow's scrollbars appropriately

---

<sup>w</sup>`xRemotelyScrolledTreeCtrl::AdjustRemoteScrollbars`

<sup>w</sup>`xremotelyscrolledtreectrladjustremotescrollbars`

<sup>b</sup>`rowse00071`

<sup>K</sup> `wxRemotelyScrolledTreeCtrl AdjustRemoteScrollbars`

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

<sup>K</sup> `AdjustRemoteScrollbars`

**wxRemotelyScrolledTreeCtrl::CalcTreeSize**

**void CalcTreeSize(const wxTreeItemId& *id*, wxRect& *rect*)<sup>K</sup>**

**void CalcTreeSize(wxRect& *rect*)<sup>K</sup>**

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

---

<sup>w</sup>xRemotelyScrolledTreeCtrl::CalcTreeSize

<sup>w</sup>xremotelyscrolledtreectrlcalctreesize

<sup>b</sup>rowse00072

<sup>K</sup> wxRemotelyScrolledTreeCtrl CalcTreeSize

<sup>E</sup>nableButton("Up");ChangeButtonBinding("Up", "JumpId(\ gizmos.hlp',

<sup>`</sup>wxremotelyscrolledtreectrl')

<sup>K</sup> CalcTreeSize

<sup>K</sup> CalcTreeSize

`$#+KK!wxRemotelyScrolledTreeCtrl::GetCompanionWindow`

`wxWindow* GetCompanionWindow() const`

---

`wxRemotelyScrolledTreeCtrl::GetCompanionWindow`

`wxremotelyscrolledtreectrlgetcompanionwindow`

`browse00073`

`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

---

`wxRemotelyScrolledTreeCtrl::GetScrolledWindow`

`wxremotelyscrolledtreectrlgetscrolledwindow`

`rowse00075`

`K wxRemotelyScrolledTreeCtrl GetScrolledWindow`

`K GetScrolledWindow`

`EnableButton("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
browse00076
K wxRemotelyScrolledTreeCtrl GetViewStart
K GetViewStart
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(^gizmos.hlp',
`wxremotelyscrolledtreectrl')")
```

**\$#+K! wxRemotelyScrolledTreeCtrl::HideVScrollbar**

**void HideVScrollbar()**<sup>K</sup>

Helpers

---

<sup>w</sup>xRemotelyScrolledTreeCtrl::HideVScrollbar

<sup>w</sup>xremotelyscrolledtreectrlhidevscrollbar

<sup>b</sup>rowse00077

<sup>K</sup> wxRemotelyScrolledTreeCtrl HideVScrollbar

<sup>E</sup>nableButton("Up");ChangeButtonBinding("Up", "JumpId(^gizmos.hlp',

`wxremotelyscrolledtreectrl')")

<sup>K</sup> HideVScrollbar

**`wxRemotelyScrolledTreeCtrl::OnExpand`**

**`void OnExpand(wxTreeEvent& event)`**<sup>K</sup>

---

`wxRemotelyScrolledTreeCtrl::OnExpand`

`wxremotelyscrolledtreectrlonexpand`

`rowse00078`

`wxRemotelyScrolledTreeCtrl OnExpand`

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

`OnExpand`



**`wxRemotelyScrolledTreeCtrl::OnPaint`**

**`void OnPaint(wxPaintEvent& event)`**<sup>K</sup>

---

<sup>w</sup>`xRemotelyScrolledTreeCtrl::OnPaint`

<sup>w</sup>`xremotelyscrolledtreectrlonpaint`

<sup>b</sup>`rowse00079`

<sup>K</sup> `wxRemotelyScrolledTreeCtrl OnPaint`

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

<sup>K</sup> `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*)**<sup>K</sup>

Events

---

<sup>w</sup>xRemotelyScrolledTreeCtrl::OnSize

<sup>w</sup>xremotelyscrolledtreectrlonsize

<sup>b</sup>rowse00081

<sup>K</sup> wxRemotelyScrolledTreeCtrl OnSize

<sup>E</sup>nableButton("Up");ChangeButtonBinding("Up", "JumpId(^gizmos.hlp',

`wxremotelyscrolledtreectrl')")

<sup>K</sup> 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*)<sup>K</sup>

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

---

<sup>w</sup>xRemotelyScrolledTreeCtrl::ScrollToLine

<sup>w</sup>xremotelyscrolledtreectrlscrolltoline

<sup>b</sup>rowse00083

<sup>K</sup> wxRemotelyScrolledTreeCtrl ScrollToLine

<sup>E</sup>nableButton("Up");ChangeButtonBinding("Up", "JumpId(^gizmos.hlp',

`wxremotelyscrolledtreectrl')")

<sup>K</sup> ScrollToLine

<sup>\$#+K!</sup>**wxRemotelyScrolledTreeCtrl::SetCompanionWindow**

**void SetCompanionWindow(wxWindow\* *companion*)**<sup>K</sup>

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

---

<sup>w</sup>xRemotelyScrolledTreeCtrl::SetCompanionWindow

<sup>w</sup>xremotelyscrolledtreectrlsetcompanionwindow

<sup>b</sup>rowse00084

<sup>K</sup> wxRemotelyScrolledTreeCtrl SetCompanionWindow

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

<sup>K</sup> SetCompanionWindow

**wxRemotelyScrolledTreeCtrl::SetScrollbars**

**void SetScrollbars**(*int pixelsPerUnitX*, *int pixelsPerUnitY*, *int noUnitsX*, *int noUnitsY*,  
*int xPos = 0*, *int yPos = 0*, **bool noRefresh = FALSE**)<sup>K</sup>

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

---

<sup>w</sup>xRemotelyScrolledTreeCtrl::SetScrollbars

<sup>w</sup>xremotelyscrolledtreectrlsetscrollbars

<sup>b</sup>rowse00085

<sup>K</sup> wxRemotelyScrolledTreeCtrl SetScrollbars

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

<sup>K</sup> SetScrollbars

<sup>\$#+K!</sup>**wxSplitterScrolledWindow::wxSplitterScrolledWindow**

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

---

<sup>w</sup>xSplitterScrolledWindow::wxSplitterScrolledWindow  
<sup>w</sup>xsplitterscrolledwindowwxsplitterscrolledwindow  
<sup>b</sup>rowse00087  
<sup>K</sup> wxSplitterScrolledWindow wxSplitterScrolledWindow  
<sup>E</sup>nableButton("Up");ChangeButtonBinding("Up", "JumpId(^gizmos.hlp',  
`wxsplitterscrolledwindow')")  
<sup>K</sup> wxSplitterScrolledWindow



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

**void OnScroll(wxScrollWinEvent& *event*)**<sup>K</sup>

Overrides Events

---

<sup>w</sup>xSplitterScrolledWindow::OnScroll

<sup>w</sup>xsplitterscrolledwindowonscroll

<sup>b</sup>rowse00088

<sup>K</sup> wxSplitterScrolledWindow OnScroll

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

<sup>K</sup> OnScroll

**`wxSplitterScrolledWindow::OnSize`**

**`void OnSize(wxSizeEvent& event)`**<sup>K</sup>

---

<sup>w</sup>`wxSplitterScrolledWindow::OnSize`

<sup>w</sup>`wxsplitterscrolledwindow::OnSize`

<sup>b</sup>`rowse00089`

<sup>K</sup> `wxSplitterScrolledWindow OnSize`

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

<sup>K</sup> `OnSize`

<sup>\$#+K!</sup>**wxThinSplitterWindow::wxThinSplitterWindow**

**wxThinSplitterWindow**(**wxWindow\*** *parent*, **wxWindowID** *id* = -1, **const wxPoint&** *pos* = *wxDefaultPosition*, **const wxSize&** *sz* = *wxDefaultSize*, **long style** = **wxSP\_3D | wxCLIP\_CHILDREN**)<sup>K</sup>

---

<sup>w</sup>xThinSplitterWindow::wxThinSplitterWindow  
<sup>w</sup>xthinsplitterwindowwxthinsplitterwindow  
<sup>b</sup>rowse00091  
<sup>K</sup> wxThinSplitterWindow wxThinSplitterWindow  
<sup>E</sup>nableButton("Up");ChangeButtonBinding("Up", "JumpId(^gizmos.hlp',  
`wxthinsplitterwindow')")  
<sup>K</sup> wxThinSplitterWindow

`$#+K! wxThinSplitterWindow::DrawSash`

`void DrawSash(wxDC& dc)K`

---

`wxThinSplitterWindow::DrawSash`

`wxthinsplitterwindowdrawsash`

`browse00092`

`K wxThinSplitterWindow DrawSash`

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

``wxthinsplitterwindow')")`

`K DrawSash`

`$#+K! wxThinSplitterWindow::OnSize`

`void OnSize(wxSizeEvent& event)K`

Events

---

`wxThinSplitterWindow::OnSize`

`wxthinsplitterwindow::OnSize`

`rowse00093`

`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()**<sup>K</sup>

Overrides

---

`w`xThinSplitterWindow::SizeWindows

`w`xthinsplitterwindowssizewindows

`b`rowse00095

`K` wxThinSplitterWindow SizeWindows

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

```wxthinsplitterwindow')")

`K` SizeWindows

<sup>\$#+K!</sup>**wxTreeCompanionWindow::wxTreeCompanionWindow**

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

---

<sup>w</sup>xTreeCompanionWindow::wxTreeCompanionWindow  
<sup>w</sup>xtreecompanionwindowwxtreecompanionwindow  
<sup>b</sup>rowse00097  
<sup>K</sup> wxTreeCompanionWindow wxTreeCompanionWindow  
<sup>E</sup>nableButton("Up");ChangeButtonBinding("Up", "JumpId(^gizmos.hlp',  
`wxtreecompanionwindow')")  
<sup>K</sup> wxTreeCompanionWindow



**wxTreeCompanionWindow::DrawItem**

**void DrawItem(wxDC& *dc*, wxTreeItemId *id*, const wxRect& *rect*)**<sup>K</sup>

Overrides

---

<sup>w</sup>xTreeCompanionWindow::DrawItem

<sup>w</sup>xtreecompanionwindowdrawitem

<sup>b</sup>rowse00098

<sup>K</sup> wxTreeCompanionWindow DrawItem

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

<sup>K</sup> DrawItem

`$#+KK!`  
**wxTreeCompanionWindow::GetTreeCtrl**  
**wxRemotelyScrolledTreeCtrl\* GetTreeCtrl() const**  
Operations Accessors

---

`w`xTreeCompanionWindow::GetTreeCtrl  
`w`xtreecompanionwindowgettreectrl  
`b`rowse00099  
`K` wxTreeCompanionWindow GetTreeCtrl  
`K` GetTreeCtrl  
`E`nableButton("Up");ChangeButtonBinding("Up", "JumpId(`gizmos.hlp',  
`wxtreecompanionwindow')")

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

**`void OnExpand(wxTreeEvent& event)K`**

---

<sup>w</sup>xTreeCompanionWindow::OnExpand

<sup>w</sup>xtreecompanionwindowonexpand

<sup>b</sup>rowse00100

<sup>K</sup> wxTreeCompanionWindow OnExpand

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

<sup>K</sup> OnExpand

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

**`void OnPaint(wxPaintEvent& event)K`**

Events

---

<sup>w</sup>xTreeCompanionWindow::OnPaint

<sup>w</sup>xtreecompanionwindowonpaint

<sup>b</sup>rowse00101

<sup>K</sup> wxTreeCompanionWindow OnPaint

<sup>E</sup>nableButton("Up");ChangeButtonBinding("Up", "JumpId(`gizmos.hlp',  
`wxtreecompanionwindow')")

<sup>K</sup> OnPaint

<sup>\$#+K!</sup>**wxTreeCompanionWindow::OnScroll**

**void OnScroll(wxScrollWinEvent& *event*)**<sup>K</sup>

---

<sup>w</sup>xTreeCompanionWindow::OnScroll

<sup>w</sup>xtreecompanionwindowonscroll

<sup>b</sup>rowse00102

<sup>K</sup> wxTreeCompanionWindow OnScroll

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

<sup>K</sup> OnScroll

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

**`void SetTreeCtrl(wxRemotelyScrolledTreeCtrl* treeCtrl)K`**

---

<sup>w</sup>`xTreeCompanionWindow::SetTreeCtrl`

<sup>w</sup>`xtreecompanionwindowsettreectrl`

<sup>b</sup>`rowse00103`

<sup>K</sup> `wxTreeCompanionWindow SetTreeCtrl`

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

<sup>K</sup> `SetTreeCtrl`







