

#\$+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](#)

---

<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.

## **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](#)

---

<sup>w</sup>xDynamicSashSplitEvent

<sup>w</sup>xdynamicsashsplitevent

<sup>b</sup>rowse00006

<sup>K</sup> wxDynamicSashSplitEvent

<sup>E</sup>nableButton("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](#)

---

<sup>w</sup>xDynamicSashUnifyEvent

<sup>w</sup>xdynamicsshunifyevent

<sup>b</sup>rowse00009

<sup>K</sup> wxDynamicSashUnifyEvent

<sup>E</sup>nableButton("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](#)

---

<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)")

\$#+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](#)

---

<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

---

`w` xMultiCellCanvas  
`w` xmulticellcanvas  
`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

---

<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](#)

---

<sup>w</sup>[wxRemotelyScrolledTreeCtrl](#)

<sup>w</sup>[xremotelyscrolledtreectrl](#)

<sup>b</sup>[rowse00068](#)

<sup>K</sup> [wxRemotelyScrolledTreeCtrl](#)

<sup>E</sup>[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)")

## \$#+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.

---

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

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

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

---

**`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  
wxdynamicssashspliteventclone  
browse00008  
K wxDynamicSashSplitEvent Clone  
K Clone  
E nableButton("Up");ChangeButtonBinding("Up", "JumpId(^gizmos.hlp', `wxdynamicssashsplitevent')")`

**`$#+K!wxDynamicSashUnifyEvent::wxDynamicSashUnifyEvent`**

**`wxDynamicSashUnifyEvent(const wxDynamicSashUnifyEvent& event)K`**

**`wxDynamicSashUnifyEvent(wxObject* target)K`**

**`wxDynamicSashUnifyEvent()K`**

---

`wxDynamicSashUnifyEvent::wxDynamicSashUnifyEvent`  
`wxdynamicsashunifyeventwxdynamicsashunifyevent`  
`browse00010`  
`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')")`

<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>xdynamicssashwindowwxdynamicssashwindow  
<sup>b</sup>rowse00013  
<sup>K</sup> wxDynamicSashWindow wxDynamicSashWindow  
<sup>E</sup>nableButton("Up");ChangeButtonBinding("Up", "JumpId(`gizmos.hlp', `wxdynamicssashwindow')")  
<sup>K</sup> wxDynamicSashWindow  
<sup>K</sup> wxDynamicSashWindow

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

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

---

<sup>w</sup>xDynamicSashWindow::~~wxDynamicSashWindow  
<sup>w</sup>xdynamicsashwindowdctor  
<sup>b</sup>rowse00014  
<sup>K</sup> wxDynamicSashWindow ~wxDynamicSashWindow  
<sup>E</sup>nableButton("Up");ChangeButtonBinding("Up", "JumpId(^ gizmos.hlp', `wxdynamicsashwindow')")  
<sup>K</sup> ~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`**

---

<sup>w</sup>xDynamicSashWindow::Create  
<sup>w</sup>xdynamicssashwindowcreate  
<sup>b</sup>rowse00016  
<sup>K</sup> wxDynamicSashWindow Create  
<sup>E</sup>nableButton("Up");ChangeButtonBinding("Up", "JumpId(`gizmos.hlp', `wxdynamicssashwindow')")  
<sup>K</sup> Create



`$#+KKl wxDynamicSashWindow::GetHScrollBar`

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

---

`w_xDynamicSashWindow::GetHScrollBar  
w_xdynamicsashwindowgethscrollbar  
b_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  
b_rowse00018  
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`**

---

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

**`$#+K! wxEditableListBox::GetStrings`**

**`void GetStrings(wxArrayString& strings)K`**

---

<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

**`$#+K! wxEditableListBox::OnDelItem`**

**`void OnDelItem(wxCommandEvent& event)K`**

---

<sup>w</sup>xEditableListBox::OnDelItem  
<sup>w</sup>xeditablelistboxondelitem  
<sup>b</sup>rowse00022  
<sup>K</sup> wxEditableListBox OnDelItem  
<sup>E</sup>nableButton("Up");ChangeButtonBinding("Up", "JumpId(^gizmos.hlp', `wxeditablelistbox')")  
<sup>K</sup> OnDelItem

**`$#+K! wxEditableListBox::OnDownItem`**

**`void OnDownItem(wxCommandEvent& event)K`**

---

<sup>w</sup>xEditableListBox::OnDownItem  
<sup>w</sup>xeditablelistboxondownitem  
<sup>b</sup>rowse00023  
<sup>K</sup> wxEditableListBox OnDownItem  
<sup>E</sup>nableButton("Up");ChangeButtonBinding("Up", "JumpId(^ gizmos.hlp', `wxeditablelistbox')")  
<sup>K</sup> OnDownItem

**`$#+K! wxEditableListBox::OnEditItem`**

**`void OnEditItem(wxCommandEvent& event)K`**

---

<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

**`$#+K!wxEditableListBox::OnEndLabelEdit`**

**`void OnEndLabelEdit(wxListEvent& event)K`**

---

<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! wxEditableListBox::OnItemSelected`**

**`void OnItemSelected(wxListEvent& event)K`**

---

<sup>w</sup>xEditableListBox::OnItemSelected  
<sup>w</sup>xeditablelistboxonitemselected  
<sup>b</sup>rowse00026  
<sup>K</sup> wxEditableListBox OnItemSelected  
<sup>E</sup>nableButton("Up");ChangeButtonBinding("Up", "JumpId(^gizmos.hlp', `wxeditablelistbox')")  
<sup>K</sup> OnItemSelected

**`$#+K! wxEditableListBox::OnNewItem`**

**`void OnNewItem(wxCommandEvent& event)K`**

---

<sup>w</sup>xEditableListBox::OnNewItem  
<sup>w</sup>xeditablelistboxonnewitem  
<sup>b</sup>rowse00027  
<sup>K</sup> wxEditableListBox OnNewItem  
<sup>E</sup>nableButton("Up");ChangeButtonBinding("Up", "JumpId(`gizmos.hlp', `wxeditablelistbox')")  
<sup>K</sup> 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)`**<sup>K</sup>

---

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

<sup>w</sup>`xeditablelistboxsetstrings`

<sup>b</sup>`rowse00029`

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

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

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

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

Constructors.

---

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

**wxLEDNumberCtrl::Create**

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

Create functions.

---

<sup>w</sup>xLEDNumberCtrl::Create  
<sup>w</sup>xlednumberctrlcreate  
<sup>b</sup>rowse00032  
<sup>K</sup> wxLEDNumberCtrl Create  
<sup>E</sup>nableButton("Up");ChangeButtonBinding("Up", "JumpId(`gizmos.hlp', `wxlednumberctrl')")  
<sup>K</sup> Create

`$#+KK!wxLEDNumberCtrl::GetAlignment`

`wxLEDValueAlign GetAlignment() const`

---

`wxLEDNumberCtrl::GetAlignment`

`wxlednumberctrlgetalignment`

`rowse00033`

`K wxLEDNumberCtrl GetAlignment`

`K GetAlignment`

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

`$#+KKl 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*)**<sup>K</sup>

---

WxLEDNumberCtrl::SetAlignment  
wxlednumberctrlsetalignment  
browse00036  
K wxLEDNumberCtrl SetAlignment  
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`gizmos.hlp', `wxlednumberctrl')")  
K 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

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

**void SetValue**(const wxString& *Value*, bool *Redraw* = *TRUE*)<sup>K</sup>

---

<sup>w</sup>xLEDNumberCtrl::SetValue  
<sup>w</sup>xlednumberctrlsetvalue  
<sup>b</sup>rowse00038  
<sup>K</sup> wxLEDNumberCtrl SetValue  
<sup>E</sup>nableButton("Up");ChangeButtonBinding("Up", "JumpId(`gizmos.hlp', `wxlednumberctrl')")  
<sup>K</sup> SetValue

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

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

---

$w$  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*)<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()K
```

---

```
wxMultiCellCanvas::MaxCols
wxmulticellcanvasmaxcols
browse00043
K wxMultiCellCanvas MaxCols
E nableButton("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`**

---

<sup>w</sup>xMultiCellCanvas::SetMinCellSize  
<sup>w</sup>xmulticellcanvassetmincellsize  
<sup>b</sup>rowse00046  
<sup>K</sup> wxMultiCellCanvas SetMinCellSize  
<sup>E</sup>nableButton("Up");ChangeButtonBinding("Up", "JumpId(`gizmos.hlp', `wxmulticellcanvas')")  
<sup>K</sup> SetMinCellSize

**`$#+K!` wxMultiCellItemHandle::wxMultiCellItemHandle**

**`wxMultiCellItemHandle(int row, int column, wxSize size, wxResizable style = wxNOT_RESIZABLE, wxSize weight = wxSize(1, 1), int align = wxALIGN_NOT)`**<sup>K</sup>

**`wxMultiCellItemHandle(int row, int column, wxResizable style, wxSize weight = wxSize(1, 1), int align = wxALIGN_NOT)`**<sup>K</sup>

**`wxMultiCellItemHandle(int row, int column, int align)`**<sup>K</sup>

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

---

<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`  
`E nableButton("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()K
```

---

```
w_xMultiCellItemHandle::GetRow
w_xmulticellitemhandlegetrow
b_rowse00053
K wxMultiCellItemHandle GetRow
E_nableButton("Up");ChangeButtonBinding("Up", "JumpId(`gizmos.hlp', `wxmulticellitemhandle')")
K GetRow
```

**$\$#+K!$  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`

---

`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**

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

---

<sup>w</sup>xMultiCellSizer::~wxMultiCellSizer

<sup>w</sup>xmulticellsizerdtor

<sup>b</sup>rowse00059

<sup>K</sup> wxMultiCellSizer ~wxMultiCellSizer

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

<sup>K</sup> ~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`

`rowse00061`

`K wxMultiCellSizer EnableGridLines`

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

`$#+K! wxMultiCellSizer::SetColumnWidth`

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

---

`w_xMultiCellSizer::SetColumnWidth  
w_xmulticellsizersetcolumnwidth  
b_rowse00064  
K wxMultiCellSizer SetColumnWidth  
E nableButton("Up");ChangeButtonBinding("Up", "JumpId(`gizmos.hlp', `wxmulticellsizer')")  
K SetColumnWidth`

`$#+K!wxMultiCellSizer::SetDefaultCellSize`

`bool SetDefaultCellSize(wxSize size)K`

---

`wxMultiCellSizer::SetDefaultCellSize`  
`wxmulticellsizersetdefaultcellsize`  
`rowse00065`  
`K wxMultiCellSizer SetDefaultCellSize`  
`E nableButton("Up");ChangeButtonBinding("Up", "JumpId(^gizmos.hlp',`wxmulticellsizer')")`  
`K SetDefaultCellSize`

**`$#+K! wxMultiCellSizer::SetGridPen`**

**`bool SetGridPen(wxPen* pen)K`**

---

`w` wxMultiCellSizer::SetGridPen  
`w` wxmulticellsizersetgridpen  
`b` rowse00066  
`K` wxMultiCellSizer SetGridPen  
`E` nableButton("Up");ChangeButtonBinding("Up", "JumpId(`gizmos.hlp', `wxmulticellsizer')")  
`K` SetGridPen

`$#+K! wxMultiCellSizer::SetRowHeight`

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

---

`w_xMultiCellSizer::SetRowHeight  
w_xmulticellsizersetrowheight  
b_rowse00067  
K wxMultiCellSizer SetRowHeight  
E nableButton("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)<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

**\$#+K!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>`wxRemotelyScrolledTreeCtrl::AdjustRemoteScrollbars`

<sup>w</sup>`wxremotelyscrolledtreectrladjustremotescrollbars`

<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', `wxremotelyscrolledtreectrl')")

<sup>K</sup> CalcTreeSize

<sup>K</sup> 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

---

<sup>w</sup>xRemotelyScrolledTreeCtrl::GetScrolledWindow  
<sup>w</sup>xremotelyscrolledtreectrlgetscrolledwindow  
<sup>b</sup>rowse00075  
<sup>K</sup> wxRemotelyScrolledTreeCtrl GetScrolledWindow  
<sup>K</sup> GetScrolledWindow  
<sup>E</sup>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()**<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

**`$#+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`**

---

<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`**

---

`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*)**<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.

---

`w_xRemotelyScrolledTreeCtrl::PrepareDC`

`w_xremotelyscrolledtreectrlpreparedc`

`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>`wxRemotelyScrolledTreeCtrl::ScrollToLine`

<sup>w</sup>`wxremotelyscrolledtreectrlscrolltoline`

<sup>b</sup>`rowse00083`

<sup>K</sup>`wxRemotelyScrolledTreeCtrl ScrollToLine`

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

<sup>K</sup>`ScrollToLine`

**\$#+K! 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> wxRemotelyScrolledTreeCtrl::SetCompanionWindow  
<sup>w</sup> wxremotelyscrolledtreectrlsetcompanionwindow  
<sup>b</sup> rowse00084  
<sup>K</sup> wxRemotelyScrolledTreeCtrl SetCompanionWindow  
<sup>E</sup> nableButton("Up");ChangeButtonBinding("Up", "JumpId(^gizmos.hlp', `wxremotelyscrolledtreectrl')")  
<sup>K</sup> SetCompanionWindow

<sup>S#+K!</sup>**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>wxRemotelyScrolledTreeCtrl::SetScrollbars

<sup>w</sup>wxremotelyscrolledtreectrlsetscrollbars

<sup>b</sup>rowse00085

<sup>K</sup>wxRemotelyScrolledTreeCtrl SetScrollbars

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

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



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

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

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

---

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

<sup>w</sup>`splitterscrolledwindow::size`

<sup>b</sup>`rowse00089`

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

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

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

Events

---

<sup>w</sup>xThinSplitterWindow::OnSize

<sup>w</sup>xthinsplitterwindowonsize

<sup>b</sup>rowse00093

<sup>K</sup> wxThinSplitterWindow OnSize

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

<sup>K</sup> OnSize

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

**bool SashHitTest**(int *x*, int *y*, int *tolerance* = 2)<sup>K</sup>

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()**<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

**\$#+K! 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', `wtreecompanionwindow')")  
<sup>K</sup> DrawItem

**\$#+KKl 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`**

---

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

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', `wtreecompanionwindow')")  
<sup>K</sup> 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`**

---

<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', `wtreecompanionwindow')")  
<sup>K</sup> SetTreeCtrl







