#$+K!

FL: a Frame Layout Framework

by Aleksandras Gluchovas and others

January 2nd 2002

Contents

Copyright notice
Introduction
Alphabetical class reference
Classes by category
Topic overviews

Contents
Contents
browse00001
K Contents
DisableButton("Up")

$#+K!Copyright notice

FL is copyright Aleksandras Gluchovas, 2001-2002.

The licence is the wxWindows Licence.

Copyright notice
topic0
browse00002
K Copyright notice
DisableButton("Up")

$#+K!**Introduction**

What is FL?
Compiling and using FL
FL concepts
Controlling dragging behaviour

I ntroduction
i ntroduction
b rowse00003
K Introduction
D isableButton("Up")

$#+K!Alphabetical class reference

BagLayout
wxBarIterator
cbAntiflickerPlugin
cbBarDimHandlerBase
cbBarDragPlugin
cbBarHintsPlugin
cbBarInfo
cbBarShapeData
cbBarSpy
cbCloseBox
cbCollapseBox
cbCommonPaneProperties
cbCustomizeBarEvent
cbCustomizeLayoutEvent
cbDimInfo
cbDockBox
cbDockPane
cbDrawBarDecorEvent
cbDrawBarHandlesEvent
cbDrawHintRectEvent
cbDrawPaneBkGroundEvent
cbDrawPaneDecorEvent
cbDrawRowBkGroundEvent
cbDrawRowDecorEvent
cbDrawRowHandlesEvent
cbDynToolBarDimHandler
cbFinishDrawInAreaEvent
cbFloatedBarWindow
cbGCUpdatesMgr
cbHintAnimationPlugin
cbInsertBarEvent
cbLayoutRowEvent
cbLayoutRowsEvent
cbLeftDClickEvent
cbLeftDownEvent
cbLeftUpEvent
cbMiniButton
cbMotionEvent
cbPaneDrawPlugin
cbPluginBase
cbPluginEvent

Alphabetical class reference
classref
browse00008
K Alphabetical class reference
DisableButton("Up")

cbRemoveBarEvent
cbResizeBarEvent
cbResizeRowEvent
cbRightDownEvent
cbRightUpEvent
cbRowDragPlugin
cbRowInfo
cbRowLayoutPlugin
cbSimpleCustomizationPlugin
cbSimpleUpdatesMgr
cbSizeBarWndEvent
cbStartBarDraggingEvent
cbStartDrawInAreaEvent
cbUpdateMgrData
cbUpdatesManagerBase
wxDynamicToolBar
wxDynToolInfo
wxFrameLayout
wxFrameManager
GarbageCollector
LayoutManagerBase
wxNewBitmapButton
wxToolLayoutItem
wxToolWindow

$#+K!**Classes by category**

A classification of FL classes by category.

Plugin classes

Plugins can be added to frame layouts to extend behaviour.

cbAntiflickerPlugin	Double-buffering class
cbBarDragPlugin	Implements drag behaviour.
cbBarHintsPlugin	Implements bar decoration and sizing behaviour.
cbHintAnimationPlugin	Draws animated hints when the user drags a pane.
cbPaneDrawPlugin	Implements most of MFC-style control bar implementation.
cbPluginBase	Abstract base class for all control-bar related plugins.
cbRowDragPlugin	Implements row-dragging functionality.
cbRowLayoutPlugin	Implements row layout functionality.
cbSimpleCustomizationPlugin	Enables customization of a bar.
cbBarSpy	Helper class used for spying for unhandled mouse events on control bars and forwarding them to the frame layout.

Window classes

Windows classes (note that the mini-button implementations are not true windows in that they do not derive from wxWindow).

wxToolWindow	A small frame that paints its own titlebar.
cbFloatedBarWindow	A kind of wxToolWindow implementing floating windows.
cbMiniButton	Base class for wxToolWindow titlebar buttons.
cbCloseBox	Close button for wxToolWindow titlebar.
cbCollapseBox	Collapse button for wxToolWindow titlebar.

Classes by category
classesbycat
browse00609
K Classes by category
DisableButton("Up")

cbDockBox	Dock button for wxToolWindow titlebar.
cbCloseBox	Close button for wxToolWindow titlebar.
wxNewBitmapButton	Alternative bitmap button class.

Layout management classes

These classes relate to the layout management framework.

cbDockPane	Manages containment and control of bars in a prent frame.
BagLayout	BagLayout lays out items in left-to-right order from top to bottom.
cbUpdatesManagerBase	An abstract interface for display update optimization logic.
cbSimpleUpdatesMgr	Implements optimized logic for refreshing areas of frame layout that need to be updated.
cbGCUpdatesMgr	Implements optimized logic for refresh, based on a garbage collection algorithm.
GarbageCollector	A garbage collection algorithm for use in display refresh optimization.
wxFrameLayout	Manages containment and docking of control bars, which can be docked along the top, bottom, right, or left side of the parent frame.

Event classes

Events are used to decouple parts of the layout framework. For event macros and identifiers, please see the topic Event macros and identifiers.

cbCustomizeBarEvent	Class for bar customization events.
cbCustomizeLayoutEvent	Class for layout customization events.
cbDrawBarDecorEvent	Class for bar decoration drawing events.
cbDrawBarHandlesEvent	Class for bar handles drawing events.
cbDrawHintRectEvent	Class for hint-rectangle drawing events.
cbDrawPaneBkGroundEvent	Class for pane background drawing events.
cbDrawPaneDecorEvent	Class for pane decoration drawing events.
cbDrawRowBkGroundEvent	Class for row background drawing events.
cbDrawRowDecorEvent	Class for row decoration drawing events.

cbDrawRowHandlesEvent	Class for row handles drawing events.
cbFinishDrawInAreaEvent	Class for finish drawing in area events.
cbInsertBarEvent	Class for bar insertion events.
cbLayoutRowEvent	Class for single row layout events.
cbLayoutRowsEvent	Class for multiple rows layout events.
cbLeftDClickEvent	Class for mouse left double click events.
cbLeftDownEvent	Class for mouse left down events.
cbLeftUpEvent	Class for mouse left up events.
cbMotionEvent	Class for mouse motion events.
cbPluginEvent	Base class for all control-bar plugin events.
cbRemoveBarEvent	Class for bar removal events.
cbResizeBarEvent	Class for bar resize events.
cbResizeRowEvent	Class for row resize events.
cbRightDownEvent	Class for mouse right down events.
cbRightUpEvent	Class for mouse right up events.
cbSizeBarWndEvent	Class for bar window resize events.
cbStartBarDraggingEvent	Class for start-bar-dragging events.
cbStartDrawInAreaEvent	Class for start drawing in area events.

$#+K!**Topic overviews**

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

Notes on using the reference
Event macros and identifiers
FAQ

Topic overviews
overviews
browse00610
K Topic overviews
DisableButton("Up")

$#+K!**What is FL?**

This manual describes FL (Frame Layout), a class library for managing sophisticated window layout, with panes that can be moved around the main window and customized. FL handles many decoration and dragging issues, giving applications the kind of docking facilities that Visual C++ and Netscape Navigator possess.

FL was written by Aleksandras Gluchovas, and is heavily used in wxWorkshop which he also wrote the bulk of.

Please note that this guide is in its infancy, and contributions from FL users are very welcome.

The following screenshot (from fl_demo1) shows a frame with a number of bars that can be dragged around. The vertical grippers with two lines allow a bar to be dragged in that row, changing the ordering of the bar if necessary. The dotted grippers (as in Netscape Navigator) allow a whole row to be moved, again changing the position of the row if required. While moving a bar or row, immediate feedback is given as the moving bar displaces other bars.

Other features: the splitter bar shows a dotted thick line as it's dragged. Single-clicking on a row handle minimizes it to a horizontal tab which is given its own narrow row. This allows the user to temporarily hide a row while allowing quick access to it when required.

A close button (x) hides a bar completely. You can get it back again by right-clicking and selecting the appropriate menu item.

A left, right, up or down arrow button expands the pane in that direction.

{bmc screen01.bmp}

Whatisfl
whatisfl
browse00004
K What is FL?
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `introduction')")

$#+K!**Compiling and using FL**

FL can be found under the 'contrib' hierarchy, in the following directories:

```
contrib/src/fl
contrib/include/wx/fl
contrib/samples/fl
contrib/docs/latex/fl
docs/html/fl
docs/htmlhelp/fl.chm
docs/pdf/fl.pdf
docs/winhelp/fl.hlp
```

To compile FL:

{bmc bullet.bmp} Under Windows using VC++, open the flVC.dsw project and compile.

{bmc bullet.bmp} Under Unix, FL should be configured when you configured wxWindows. Make FL by changing directory to contrib/src/fl and type 'make'. **Note:** there is currently a problem with the wxWindows build system that means that only the static version of library can be built at present.

To use FL:

{bmc bullet.bmp} Under Windows using VC++, link with fl[d].lib.

{bmc bullet.bmp} Under Unix, link with libfl[d].a.

Compiling and using FL
topic1
browse00005
K Compiling and using FL
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `introduction')")

$#+K!**FL concepts**

These are typical steps when adding FL functionality to your application.

{bmc bullet.bmp} include the appropriate header files;

{bmc bullet.bmp} create a new wxFrameLayout passing the top-level frame and the window that is interpreted as the main 'client' window;

{bmc bullet.bmp} set an updates manager for optimizing drag operations;

{bmc bullet.bmp} add plugins for implementing various features;

{bmc bullet.bmp} add bars;

{bmc bullet.bmp} enable floating mode for the layout if required;

{bmc bullet.bmp} delete the frame layout in the main frame's destructor.

The following is taken from fl_demo1 and shows the main code implementing the user interface as illustrated in What is FL?.

```
// fl headers
#include "wx/fl/controlbar.h"     // core API

// extra plugins
#include "wx/fl/barhintspl.h"    // bevel for bars with "X"s and
grooves
#include "wx/fl/rowdragpl.h"     // NC-look with draggable rows
#include "wx/fl/cbcustom.h"      // customization plugin
#include "wx/fl/hintanimpl.h"

// beauty-care
#include "wx/fl/gcupdatesmgr.h"  // smooth d&d
#include "wx/fl/antiflickpl.h"   // double-buffered repaint of
decorations
#include "wx/fl/dyntbar.h"       // auto-layout toolbar
#include "wx/fl/dyntbarhnd.h"    // control-bar dimension handler
for it

MyFrame::MyFrame(wxFrame *frame)
    : wxFrame( frame, -1, "wxWindows 2.0 wxFrameLayout Test
Application", wxDefaultPosition,
          wxSize( 700, 500 ),
          wxCLIP_CHILDREN | wxMINIMIZE_BOX | wxMAXIMIZE_BOX |
          wxTHICK_FRAME   | wxSYSTEM_MENU  | wxCAPTION,
          "freimas" )
{
    mpClientWnd = CreateTextCtrl( "Client window" );

    mpLayout = new wxFrameLayout( this, mpClientWnd );
```

F L concepts
t opic2
b rowse00006
K FL concepts
E nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `introduction')")

```
    mpLayout->SetUpdatesManager( new cbGCUpdatesMgr() );

    // setup plugins for testing
    mpLayout->PushDefaultPlugins();

    mpLayout->AddPlugin( CLASSINFO( cbBarHintsPlugin ) ); // fancy
"X"es and bevel for bars
    mpLayout->AddPlugin( CLASSINFO( cbHintAnimationPlugin ) );
    mpLayout->AddPlugin( CLASSINFO( cbRowDragPlugin  ) );
    mpLayout->AddPlugin( CLASSINFO( cbAntiflickerPlugin ) );
    mpLayout->AddPlugin( CLASSINFO( cbSimpleCustomizationPlugin )
);

    // drop in some bars
    cbDimInfo sizes0( 200,45, // when docked horizontally
                      200,85, // when docked vertically
                      175,35, // when floated
                      FALSE,  // the bar is not fixed-size
                      4,      // vertical gap (bar border)
                      4       // horizontal gap (bar border)
                    );

    cbDimInfo sizes1( 150,35, // when docked horizontally
                      150,85, // when docked vertically
                      175,35, // when floated
                      TRUE,   // the bar is not fixed-size
                      4,      // vertical gap (bar border)
                      4       // horizontal gap (bar border)
                    );

    cbDimInfo sizes2( 175,45, // when docked horizontally
                      175,37, // when docked vertically
                      170,35, // when floated
                      TRUE,   // the bar is not fixed-size
                      4,      // vertical gap (bar border)
                      4,      // horizontal gap (bar border)
                      new cbDynToolBarDimHandler()
                    );

    mpLayout->AddBar( CreateTextCtrl("Hello"),  // bar window
                      sizes0, FL_ALIGN_TOP,     // alignment ( 0-
top,1-bottom, etc)
                      0,                        // insert into 0th
row (vert. position)
                      0,                        // offset from the
start of row (in pixels)
                      "InfoViewer1",            // name for
reference in customization pop-ups
                      TRUE
                    );

    mpLayout->AddBar( CreateTextCtrl("Bye"),    // bar window
                      sizes0, FL_ALIGN_TOP,     // alignment ( 0-
top,1-bottom, etc)
                      1,                        // insert into 0th
row (vert. position)
                      0,                        // offset from the
start of row (in pixels)
                      "InfoViewer2",            // name for
reference in customization pop-ups
                      TRUE
                    );
```

```
    mpLayout->AddBar( CreateTextCtrl("Fixed0"), // bar window
                      sizes1, FL_ALIGN_TOP,     // alignment ( 0-
top,1-bottom, etc)
                      0,                        // insert into 0th
row (vert. position)
                      0,                        // offset from the
start of row (in pixels)
                      "ToolBar1",               // name for
reference in customization pop-ups
                      TRUE
                    );

    wxDynamicToolBar* pToolBar = new wxDynamicToolBar();

    pToolBar->Create( this, -1 );

    // 1001-1006 ids of command events fired by added tool-buttons

    pToolBar->AddTool( 1001, BMP_DIR "new.bmp" );
    pToolBar->AddTool( 1002, BMP_DIR "open.bmp" );
    pToolBar->AddTool( 1003, BMP_DIR "save.bmp" );

    pToolBar->AddTool( 1004, BMP_DIR "cut.bmp" );
    pToolBar->AddTool( 1005, BMP_DIR "copy.bmp" );
    pToolBar->AddTool( 1006, BMP_DIR "paste.bmp" );

    mpLayout->AddBar( pToolBar,             // bar window (can be
NULL)
                      sizes2, FL_ALIGN_TOP, // alignment ( 0-
top,1-bottom, etc)
                      0,                    // insert into 0th row
(vert. position)
                      0,                    // offset from the
start of row (in pixels)
                      "ToolBar2",           // name for reference
in customization pop-ups
                      FALSE
                    );

    mpLayout->EnableFloating( TRUE ); // off, thinking about
wxGtk...
}

MyFrame::~MyFrame()
{
    if ( mpLayout)
        delete mpLayout; // should be destroyed manually
}
```

$#+K!Controlling dragging behaviour

Various pane-dragging behaviours are supported. FL can show an outline of where the window would be docked if you stopped dragging at that point.

This is a list of properties of interest in the cbCommonPaneProperties structure:

```
bool mRealTimeUpdatesOn;     // default: ON
bool mOutOfPaneDragOn;       // default: ON
bool mExactDockPredictionOn; // default: OFF
bool mNonDestructFrictionOn; // default: OFF
```

To get behaviour similar to Microsoft's DevStudio drag-ghost behaviour, mRealTimeUpdatesOn have to be set to FALSE, for example:

```
cbCommonPaneProperties props;
....
....
props.mRealTimeUpdatesOn = FALSE;
fl->SetPaneProperties( props, wxALL_PANES );
```

mOutOfPaneDragOn specifies whether bars can be dragged away from this pane. (Note: this may not currently be working.)

mExactDockPredictionOn is only relevant when *mRealTimeUpdatesOn* is FALSE, and then the hint rectangle behaves a little jumpily. It tries to show exatly how the bar would look and where it would be docked if the dragging finished right now, i.e. the final position, with all the 'friction-physics' calculated. Otherwise the hint flies smothly above the surface only hinting whether the bar will be docked vertically or horizontally if dropped now. This is a feature you won't find anywhere else!

mNonDestructFirctionOn causes the bars not being dragged to stay where they are, while the currently dragged one is 'diving' through the underlaying panes, docking itself in and out in real time. Otherwise the stationary bars would be pushed around messing up the composition permanently. This flag is irelevant when *mRealTimeUpdatesOn* is FALSE, as the ghost-rect does not do any docking until the drag finishes.

C ontrolling dragging behaviour
c ontrollingdragbehav
b rowse00007
K Controlling dragging behaviour
E nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `introduction')")

$#+K!**BagLayout**

BagLayout lays out items in left-to-right order from top to bottom.

Derived from

LayoutManagerBase

Include files

<wx/fl/dyntbar.h>

Data structures

Members

BagLayout::Layout

BagLayout
baglayout
browse00009
K BagLayout
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `classref')")

$#+K!**wxBarIterator**

Used for traversing through all bars of all rows in the pane.

Derived from

No base class

Include files

<wx/fl/controlbar.h>

Data structures

Members

wxBarIterator::wxBarIterator
wxBarIterator::BarInfo
wxBarIterator::Next
wxBarIterator::Reset
wxBarIterator::RowInfo

---

wxBarIterator
wxbariterator
browse00011
wxBarIterator
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `classref')")

$#+K!**cbAntiflickerPlugin**

Implements double-buffering to reduce flicker. Bitmap and memory DC buffers are shared 'resources' among all instances of antiflicker plugins within the application.

Locking for multithreaded applications is not yet implemented.

Derived from

cbPluginBase

Include files

<wx/fl/antiflickpl.h>

Data structures

Members

cbAntiflickerPlugin::cbAntiflickerPlugin
cbAntiflickerPlugin::~cbAntiflickerPlugin
cbAntiflickerPlugin::AllocNewBuffer
cbAntiflickerPlugin::FindSuitableBuffer
cbAntiflickerPlugin::GetClientDC
cbAntiflickerPlugin::GetWindowDC
cbAntiflickerPlugin::OnFinishDrawInArea
cbAntiflickerPlugin::OnStartDrawInArea

---

cbAntiflickerPlugin
cbantiflickerplugin
browse00017
K cbAntiflickerPlugin
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `classref')")

$#+K!**cbBarDimHandlerBase**

Abstract interface for bar-size handler classes. These objects receive notifications whenever the docking state of the bar is changed, thus they provide the possibility to adjust the values in cbDimInfo::mSizes accordingly. Specific handlers can be hooked up to specific types of bar.

Derived from

wxObject

Include files

<wx/fl/controlbar.h>

Data structures

Members

cbBarDimHandlerBase::cbBarDimHandlerBase
cbBarDimHandlerBase::AddRef
cbBarDimHandlerBase::OnChangeBarState
cbBarDimHandlerBase::OnResizeBar
cbBarDimHandlerBase::RemoveRef

---

cbBarDimHandlerBase
cbbardimhandlerbase
browse00026
K cbBarDimHandlerBase
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `classref')")

$#+K!**cbBarDragPlugin**

Plugin class implementing bar dragging.

Derived from

cbPluginBase

Include files

<wx/fl/bardragpl.h>

Data structures

Members

cbBarDragPlugin::cbBarDragPlugin
cbBarDragPlugin::~cbBarDragPlugin
cbBarDragPlugin::AdjustHintRect
cbBarDragPlugin::CalcOnScreenDims
cbBarDragPlugin::ClipPosInFrame
cbBarDragPlugin::ClipRectInFrame
cbBarDragPlugin::DoDrawHintRect
cbBarDragPlugin::DrawHintRect
cbBarDragPlugin::EraseHintRect
cbBarDragPlugin::FinishTracking
cbBarDragPlugin::GetBarHeightInPane
cbBarDragPlugin::GetBarWidthInPane
cbBarDragPlugin::GetDistanceToPane
cbBarDragPlugin::HitTestPanes
cbBarDragPlugin::HitsPane
cbBarDragPlugin::IsInClientArea
cbBarDragPlugin::IsInOtherPane
cbBarDragPlugin::OnDrawHintRect
cbBarDragPlugin::OnLButtonDown
cbBarDragPlugin::OnLButtonUp
cbBarDragPlugin::OnLDblClick
cbBarDragPlugin::OnMouseMove
cbBarDragPlugin::OnStartBarDragging
cbBarDragPlugin::RectToScr
cbBarDragPlugin::ShowHint
cbBarDragPlugin::StartTracking
cbBarDragPlugin::StickToPane
cbBarDragPlugin::UnstickFromPane

---

cbBarDragPlugin
cbbardragplugin
browse00032
K cbBarDragPlugin
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `classref')")

$#+K!**cbBarHintsPlugin**

This class intercepts bar-decoration and sizing events, and draws 3D hints around fixed and flexible bars, similar to those in Microsoft DevStudio 6.x

Derived from

cbPluginBase

Include files

<wx/fl/barhintspl.h>

Data structures

Members

cbBarHintsPlugin::cbBarHintsPlugin
cbBarHintsPlugin::~cbBarHintsPlugin
cbBarHintsPlugin::CreateBoxes
cbBarHintsPlugin::DoDrawHint
cbBarHintsPlugin::Draw3DBox
cbBarHintsPlugin::DrawCloseBox
cbBarHintsPlugin::DrawCollapseBox
cbBarHintsPlugin::DrawGrooves
cbBarHintsPlugin::ExcludeHints
cbBarHintsPlugin::GetHintsLayout
cbBarHintsPlugin::HitTestHints
cbBarHintsPlugin::OnDrawBarDecorations
cbBarHintsPlugin::OnInitPlugin
cbBarHintsPlugin::OnLeftDown
cbBarHintsPlugin::OnLeftUp
cbBarHintsPlugin::OnMotion
cbBarHintsPlugin::OnSizeBarWindow
cbBarHintsPlugin::SetGrooveCount

---

cbBarHintsPlugin
cbbarhintsplugin
browse00061
K cbBarHintsPlugin
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `classref')")

$#+K!**cbBarInfo**

Helper class used internally by the wxFrameLayout class. Holds and manages bar information.

Derived from

wxObject

Include files

<wx/fl/controlbar.h>

Data structures

Members

cbBarInfo::cbBarInfo
cbBarInfo::~cbBarInfo
cbBarInfo::IsExpanded
cbBarInfo::IsFixed

---

cbBarInfo
cbbarinfo
browse00080
cbBarInfo
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `classref')")

$#+K!cbBarShapeData

Used for storing the original bar's positions in the row, when the 'non-destructive-friction' option is turned on.

Derived from

wxObject

Include files

<wx/fl/controlbar.h>

Data structures

---

cbBarShapeData
cbbarshapedata
browse00085
K cbBarShapeData
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `classref')")

$#+K!**cbBarSpy**

Helper class, used for spying for unhandled mouse events on control bars and forwarding them to the frame layout.

Derived from

wxEvtHandler

Include files

<wx/fl/controlbar.h>

Data structures

```
typedef cbBarInfo* BarInfoPtrT
```

forward declarations

```
typedef cbRowInfo* RowInfoPtrT
```

enumeration of hittest results, see cbDockPane::HitTestPaneItems(..)

```
enum CB_HITTEST_RESULT
{
    CB_NO_ITEMS_HITTED,

    CB_UPPER_ROW_HANDLE_HITTED,
    CB_LOWER_ROW_HANDLE_HITTED,
    CB_LEFT_BAR_HANDLE_HITTED,
    CB_RIGHT_BAR_HANDLE_HITTED,
    CB_BAR_CONTENT_HITTED
}
```

Members

cbBarSpy::cbBarSpy
cbBarSpy::ProcessEvent
cbBarSpy::SetBarWindow

---

cbBarSpy
cbbarspy
browse00086
cbBarSpy
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `classref')")

$#+K!**cbCloseBox**

cbCloseBox is a window close button, used in a wxToolWindow titlebar.

Derived from

cbMiniButton

Include files

<wx/fl/toolwnd.h>

Data structures

Members

cbCloseBox::Draw

---

cbCloseBox
cbclosebox
browse00090
K cbCloseBox
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `classref')")

$#+K!**cbCollapseBox**

cbCollapseBox is a window collapse button, used in a wxToolWindow titlebar.

Derived from

cbMiniButton

Include files

<wx/fl/toolwnd.h>

Data structures

Members

cbCollapseBox::Draw

---

cbCollapseBox
cbcollapsebox
browse00092
K cbCollapseBox
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `classref')")

$#+K!**cbCommonPaneProperties**

A structure holding configuration options, which are usually the same for all panes in a frame layout. For an explanation of the data members, please see Controlling dragging behaviour.

Derived from

wxObject

Include files

<wx/fl/controlbar.h>

Data structures

```
class cbCommonPaneProperties : public wxObject
{
    DECLARE_DYNAMIC_CLASS( cbCommonPaneProperties )

    // Look-and-feel configuration

    bool mRealTimeUpdatesOn;     // default: ON
    bool mOutOfPaneDragOn;       // default: ON
    bool mExactDockPredictionOn; // default: OFF
    bool mNonDestructFrictionOn; // default: OFF

    bool mShow3DPaneBorderOn;    // default: ON

    // The following properties are reserved for the "future"

    bool mBarFloatingOn;         // default: OFF
    bool mRowProportionsOn;      // default: OFF
    bool mColProportionsOn;      // default: ON
    bool mBarCollapseIconsOn;    // default: OFF
    bool mBarDragHintsOn;        // default: OFF

    // Minimal dimensions for not-fixed bars in this pane (16x16
default)

    wxSize mMinCBarDim;

    // Width/height of resizing sash

    int    mResizeHandleSize;

    // Default constructor.

    cbCommonPaneProperties(void);
};
```

---

[c]bCommonPaneProperties
[c]bcommonpaneproperties
[b]rowse00094
[K] cbCommonPaneProperties
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `classref')")

$#+K!**cbCustomizeBarEvent**

Class for bar customization events.

Derived from

cbPluginEvent

Include files

<wx/fl/controlbar.h>

Data structures

Members

cbCustomizeBarEvent::cbCustomizeBarEvent

---

cbCustomizeBarEvent
cbcustomizebarevent
browse00095
K cbCustomizeBarEvent
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `classref')")

$#+K!**cbCustomizeLayoutEvent**

Class for layout customization events.

Derived from

cbPluginEvent

Include files

<wx/fl/controlbar.h>

Data structures

Members

cbCustomizeLayoutEvent::cbCustomizeLayoutEvent

---

cbCustomizeLayoutEvent
cbcustomizelayoutevent
browse00097
K cbCustomizeLayoutEvent
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `classref')")

$#+K!**cbDimInfo**

Helper class used internally by the wxFrameLayout class. Holds and manages information about bar dimensions.

Derived from

wxObject

Include files

<wx/fl/controlbar.h>

Data structures

Members

cbDimInfo::cbDimInfo
cbDimInfo::~cbDimInfo
cbDimInfo::GetDimHandler
cbDimInfo::operator=

---

cbDimInfo
cbdiminfo
browse00099
K cbDimInfo
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `classref')")

$#+K!**cbDockBox**

cbDockBox is a window dock button, used in a wxToolWindow titlebar.

Derived from

cbMiniButton

Include files

<wx/fl/toolwnd.h>

Data structures

Members

cbDockBox::Draw

---

cbDockBox
cbdockbox
browse00104
K cbDockBox
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `classref')")

$#+K!**cbDockPane**

This class manages containment and control of control bars along one of the four edges of the parent frame.

Derived from

wxObject

Include files

<wx/fl/controlbar.h>

Data structures

Members

cbDockPane::cbDockPane
cbDockPane::~cbDockPane
cbDockPane::BarPresent
cbDockPane::CalcLengthRatios
cbDockPane::ContractBar
cbDockPane::DoInsertBar
cbDockPane::DrawHorizHandle
cbDockPane::DrawVertHandle
cbDockPane::ExpandBar
cbDockPane::FinishDrawInArea
cbDockPane::FrameToPane
cbDockPane::GetAlignment
cbDockPane::GetBarInfoByWindow
cbDockPane::GetBarResizeRange
cbDockPane::GetDockingState
cbDockPane::GetFirstRow
cbDockPane::GetMinimalRowHeight
cbDockPane::GetNotFixedBarsCount
cbDockPane::GetPaneHeight
cbDockPane::GetRealRect
cbDockPane::GetRow
cbDockPane::GetRowAt
cbDockPane::GetRowIndex
cbDockPane::GetRowList
cbDockPane::GetRowResizeRange
cbDockPane::GetRowShapeData
cbDockPane::GetRowY
cbDockPane::HasNotFixedBarsLeft

---

cbDockPane
cbdockpane
browse00106
K cbDockPane
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `classref')")

cbDockPane::HasNotFixedBarsRight
cbDockPane::HasNotFixedRowsAbove
cbDockPane::HasNotFixedRowsBelow
cbDockPane::HasPoint
cbDockPane::HitTestPaneItems
cbDockPane::InitLinksForRow
cbDockPane::InitLinksForRows
cbDockPane::InsertBar
cbDockPane::InsertRow
cbDockPane::IsFixedSize
cbDockPane::IsHorizontal
cbDockPane::MatchesMask
cbDockPane::PaintBar
cbDockPane::PaintBarDecorations
cbDockPane::PaintBarHandles
cbDockPane::PaintPane
cbDockPane::PaintPaneBackground
cbDockPane::PaintPaneDecorations
cbDockPane::PaintRow
cbDockPane::PaintRowBackground
cbDockPane::PaintRowDecorations
cbDockPane::PaintRowHandles
cbDockPane::PaneToFrame
cbDockPane::RecalcLayout
cbDockPane::RecalcRowLayout
cbDockPane::RemoveBar
cbDockPane::RemoveRow
cbDockPane::ResizeBar
cbDockPane::ResizeRow
cbDockPane::SetBoundsInParent
cbDockPane::SetMargins
cbDockPane::SetPaneWidth
cbDockPane::SetRowHeight
cbDockPane::SetRowShapeData
cbDockPane::SizeBar
cbDockPane::SizePaneObjects
cbDockPane::SizeRowObjects
cbDockPane::StartDrawInArea
cbDockPane::SyncRowFlags

$#+K!**cbDrawBarDecorEvent**

Class for bar decoration drawing events.

Derived from

cbPluginEvent

Include files

<wx/fl/controlbar.h>

Data structures

Members

cbDrawBarDecorEvent::cbDrawBarDecorEvent

---

cbDrawBarDecorEvent
cbdrawbardecorevent
browse00174
K cbDrawBarDecorEvent
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `classref')")

$#+K!**cbDrawBarHandlesEvent**

Class for bar handles drawing events.

Derived from

cbPluginEvent

Include files

<wx/fl/controlbar.h>

Data structures

Members

cbDrawBarHandlesEvent::cbDrawBarHandlesEvent

---

cbDrawBarHandlesEvent
cbdrawbarhandlesevent
browse00176
K cbDrawBarHandlesEvent
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `classref')")

$#+K!**cbDrawHintRectEvent**

Class for hint-rectangle drawing events.

Derived from

cbPluginEvent

Include files

<wx/fl/controlbar.h>

Data structures

Members

cbDrawHintRectEvent::cbDrawHintRectEvent

---

cbDrawHintRectEvent
cbdrawhintrectevent
browse00178
K cbDrawHintRectEvent
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `classref')")

$#+K!**cbDrawPaneBkGroundEvent**

Class for pane background drawing events.

Derived from

cbPluginEvent

Include files

<wx/fl/controlbar.h>

Data structures

Members

cbDrawPaneBkGroundEvent::cbDrawPaneBkGroundEvent

---

cbDrawPaneBkGroundEvent
cbdrawpanebkgroundevent
browse00180
K cbDrawPaneBkGroundEvent
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `classref')")

$#+K!**cbDrawPaneDecorEvent**

Class for pane decoration drawing events.

Derived from

cbPluginEvent

Include files

<wx/fl/controlbar.h>

Data structures

Members

cbDrawPaneDecorEvent::cbDrawPaneDecorEvent

---

cbDrawPaneDecorEvent
cbdrawpanedecorevent
browse00182
K cbDrawPaneDecorEvent
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `classref')")

$#+K!**cbDrawRowBkGroundEvent**

Class for row background drawing events.

Derived from

cbPluginEvent

Include files

<wx/fl/controlbar.h>

Data structures

Members

cbDrawRowBkGroundEvent::cbDrawRowBkGroundEvent

---

cbDrawRowBkGroundEvent
cbdrawrowbkgroundevent
browse00184
K cbDrawRowBkGroundEvent
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `classref')")

$#+K!**cbDrawRowDecorEvent**

Class for row decoration drawing events.

Derived from

cbPluginEvent

Include files

<wx/fl/controlbar.h>

Data structures

Members

cbDrawRowDecorEvent::cbDrawRowDecorEvent

---

cbDrawRowDecorEvent
cbdrawrowdecorevent
browse00186
K cbDrawRowDecorEvent
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `classref')")

$#+K!**cbDrawRowHandlesEvent**

Class for row handles drawing events.

Derived from

cbPluginEvent

Include files

<wx/fl/controlbar.h>

Data structures

Members

cbDrawRowHandlesEvent::cbDrawRowHandlesEvent

---

cbDrawRowHandlesEvent
cbdrawrowhandlesevent
browse00188
K cbDrawRowHandlesEvent
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `classref')")

$#+K!**cbDynToolBarDimHandler**

Dynamic toolbar dimension handler.

Derived from

cbBarDimHandlerBase

Include files

<wx/fl/dyntbarhnd.h>

Data structures

Members

cbDynToolBarDimHandler::OnChangeBarState
cbDynToolBarDimHandler::OnResizeBar

---

cbDynToolBarDimHandler
cbdyntoolbardimhandler
browse00190
K cbDynToolBarDimHandler
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `classref')")

$#+K!**cbFinishDrawInAreaEvent**

Class for finish drawing in area events.

Derived from

cbPluginEvent

Include files

<wx/fl/controlbar.h>

Data structures

Members

cbFinishDrawInAreaEvent::cbFinishDrawInAreaEvent

---

cbFinishDrawInAreaEvent
cbfinishdrawinareaevent
browse00193
K cbFinishDrawInAreaEvent
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `classref')")

$#+K!**cbFloatedBarWindow**

cbFloatedBarWindow is a kind of wxToolWindow, implementing floating toolbars.

Derived from

wxToolWindow

Include files

<wx/fl/toolwnd.h>

Data structures

Members

cbFloatedBarWindow::cbFloatedBarWindow
cbFloatedBarWindow::GetBar
cbFloatedBarWindow::GetPreferredSize
cbFloatedBarWindow::HandleTitleClick
cbFloatedBarWindow::OnDblClick
cbFloatedBarWindow::OnMiniButtonClicked
cbFloatedBarWindow::PositionFloatedWnd
cbFloatedBarWindow::SetBar
cbFloatedBarWindow::SetLayout

---

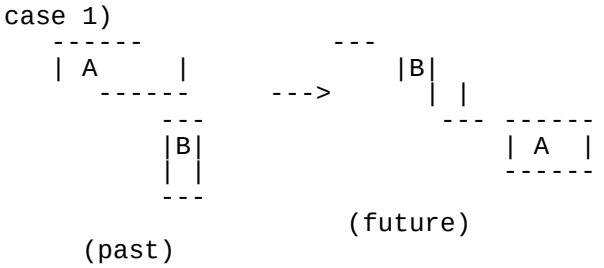
cbFloatedBarWindow
cbfloatedbarwindow
browse00195
K cbFloatedBarWindow
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `classref')")

$#+K!**cbGCUpdatesMgr**

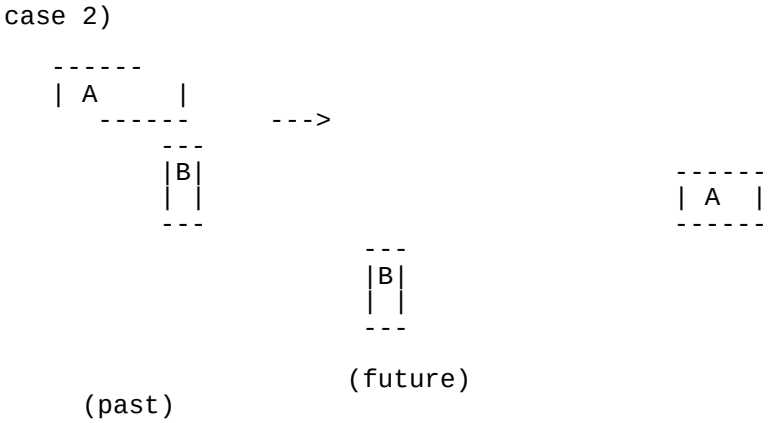
This class implements optimized logic for refreshing the areas of frame layout that actually need to be updated. It is used as the default updates manager by wxFrameLayout.

It is called 'Garbage Collecting' updates manager because its implementation tries to find out dependencies between bars, and to order them into a 'hierarchy'. This hierarchical sorting resembles the implementation of heap-garbage collectors, which resolve dependencies between references.

Example: there are situations where the order in which the user moves windows does matter.



Past/future positions of A and B windows completely overlap, i.e. depend on each other, and there is no solution for moving the windows without refreshing both of them -- we have a cyclic dependency here. The garbage collection algorithm will find this cyclic dependecy and will force refresh after movement.



In this case past/future positions do not overlap, so it is enough only to move windows without refreshing them. Garbage collection will 'notice' this.

---

cbGCUpdatesMgr
cbgcupdatesmgr
browse00205
K cbGCUpdatesMgr
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `classref')")

There is also a third case, when overlapping is partial. In this case the refreshing can also be avoided by moving windows in the order of 'most-dependant' towards the 'least-dependent'. GC handles this automatically, by sorting windows by their dependency-level (or 'hierarchy').

See garbagec.h for more details of this method; garbagec.h/cpp implement sorting of generic dependencies and does not deal with graphical objects directly.

Summary: garbage collection improves performance when complex or large windows are moved around, by reducing the number of repaints. It also helps to avoid dirty non-client areas of moved windows in some special cases of 'overlapping anomalies'.

Derived from

cbSimpleUpdatesMgr

Include files

<wx/fl/gcupdatesmgr.h>

Data structures

Members

cbGCUpdatesMgr::cbGCUpdatesMgr
cbGCUpdatesMgr::AddItem
cbGCUpdatesMgr::DoRepositionItems
cbGCUpdatesMgr::OnStartChanges
cbGCUpdatesMgr::UpdateNow

$#+K!**cbHintAnimationPlugin**

A plugin to draw animated hints when the user drags a pane.

Derived from

cbPluginBase

Include files

<wx/fl/hintanimpl.h>

Data structures

Members

cbHintAnimationPlugin::cbHintAnimationPlugin
cbHintAnimationPlugin::~cbHintAnimationPlugin
cbHintAnimationPlugin::DoDrawHintRect
cbHintAnimationPlugin::DrawHintRect
cbHintAnimationPlugin::EraseHintRect
cbHintAnimationPlugin::FinishTracking
cbHintAnimationPlugin::OnDrawHintRect
cbHintAnimationPlugin::RectToScr
cbHintAnimationPlugin::StartTracking

---

cbHintAnimationPlugin
cbhintanimationplugin
browse00211
K cbHintAnimationPlugin
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `classref')")

$#+K!**cbInsertBarEvent**

Class for bar insertion events.

Derived from

cbPluginEvent

Include files

<wx/fl/controlbar.h>

Data structures

Members

cbInsertBarEvent::cbInsertBarEvent

---

cbInsertBarEvent
cbinsertbarevent
browse00221
K cbInsertBarEvent
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `classref')")

$#+K!**cbLayoutRowEvent**

Class for single row layout events.

Derived from

cbPluginEvent

Include files

<wx/fl/controlbar.h>

Data structures

Members

cbLayoutRowEvent::cbLayoutRowEvent

---

cbLayoutRowEvent
cblayoutrowevent
browse00223
K cbLayoutRowEvent
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `classref')")

$#+K!**cbLayoutRowsEvent**

Class for multiple rows layout events.

Derived from

cbPluginEvent

Include files

<wx/fl/controlbar.h>

Data structures

Members

cbLayoutRowsEvent::cbLayoutRowsEvent

---

cbLayoutRowsEvent
cblayoutrowsevent
browse00225
cbLayoutRowsEvent
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `classref')")

$#+K!**cbLeftDClickEvent**

Class for mouse left double click events.

Derived from

cbPluginEvent

Include files

<wx/fl/controlbar.h>

Data structures

Members

cbLeftDClickEvent::cbLeftDClickEvent

---

cbLeftDClickEvent
cbleftdclickevent
browse00227
K cbLeftDClickEvent
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `classref')")

$#+K!**cbLeftDownEvent**

Class for mouse left down events.

Derived from

cbPluginEvent

Include files

<wx/fl/controlbar.h>

Data structures

Members

cbLeftDownEvent::cbLeftDownEvent

---

cbLeftDownEvent
cbleftdownevent
browse00229
K cbLeftDownEvent
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `classref')")

$#+K!**cbLeftUpEvent**

Class for mouse left up events.

Derived from

cbPluginEvent

Include files

<wx/fl/controlbar.h>

Data structures

Members

cbLeftUpEvent::cbLeftUpEvent

---

cbLeftUpEvent
cbleftupevent
browse00231
K cbLeftUpEvent
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `classref')")

$#+K!**cbMiniButton**

cbMiniButton is the base class for a small button that can be placed in a wxToolWindow titlebar.

Derived from

wxObject

Include files

<wx/fl/toolwnd.h>

Data structures

Members

cbMiniButton::cbMiniButton
cbMiniButton::Draw
cbMiniButton::Enable
cbMiniButton::HitTest
cbMiniButton::IsPressed
cbMiniButton::OnLeftDown
cbMiniButton::OnLeftUp
cbMiniButton::OnMotion
cbMiniButton::Refresh
cbMiniButton::Reset
cbMiniButton::SetPos
cbMiniButton::WasClicked

---

cbMiniButton
cbminibutton
browse00233
K cbMiniButton
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `classref')")

$#+K!**cbMotionEvent**

Class for mouse motion events.

Derived from

cbPluginEvent

Include files

<wx/fl/controlbar.h>

Data structures

Members

cbMotionEvent::cbMotionEvent

---

cbMotionEvent
cbmotionevent
browse00246
K cbMotionEvent
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `classref')")

$#+K!**cbPaneDrawPlugin**

Simple but all-in-one plugin implementation. Resembles the look and feel of to MFC control-bars. The class handles painting of the pane and the items in it; it generates bar/layout customization events, when the user right-clicks the bar/pane. Hooking an instance of this and row-layout plugins for each pane would be enough for the frame layout to function properly (they are plugged in automatically by the wxFrameLayout class).

Derived from

cbPluginBase

Include files

<wx/fl/panedrawpl.h>

Data structures

Members

cbPaneDrawPlugin::cbPaneDrawPlugin
cbPaneDrawPlugin::~cbPaneDrawPlugin
cbPaneDrawPlugin::Clone
cbPaneDrawPlugin::DrawBarInnerShadeRect
cbPaneDrawPlugin::DrawDraggedHandle
cbPaneDrawPlugin::DrawLowerRowHandle
cbPaneDrawPlugin::DrawLowerRowShades
cbPaneDrawPlugin::DrawPaneShade
cbPaneDrawPlugin::DrawPaneShadeForRow
cbPaneDrawPlugin::DrawShade
cbPaneDrawPlugin::DrawShade1
cbPaneDrawPlugin::DrawUpperRowHandle
cbPaneDrawPlugin::DrawUpperRowShades
cbPaneDrawPlugin::OnDrawBarDecorations
cbPaneDrawPlugin::OnDrawBarHandles
cbPaneDrawPlugin::OnDrawPaneBackground
cbPaneDrawPlugin::OnDrawPaneDecorations
cbPaneDrawPlugin::OnDrawRowBackground
cbPaneDrawPlugin::OnDrawRowDecorations
cbPaneDrawPlugin::OnDrawRowHandles
cbPaneDrawPlugin::OnFinishDrawInArea
cbPaneDrawPlugin::OnLButtonDown
cbPaneDrawPlugin::OnLButtonUp
cbPaneDrawPlugin::OnLDblClick

---

cbPaneDrawPlugin
cbpanedrawplugin
browse00248
K cbPaneDrawPlugin
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `classref')")

cbPaneDrawPlugin::OnMouseMove
cbPaneDrawPlugin::OnRButtonUp
cbPaneDrawPlugin::OnSizeBarWindow
cbPaneDrawPlugin::OnStartDrawInArea
cbPaneDrawPlugin::SetDarkPixel
cbPaneDrawPlugin::SetLightPixel

$#+K!**cbPluginBase**

Abstract base class for all control-bar related plugins. Note: pointer positions of mouse events sent to plugins are always in the pane's coordinates (the pane to which this plugin is hooked).

Derived from

wxEvtHandler

Include files

<wx/fl/controlbar.h>

Data structures

Members

cbPluginBase::cbPluginBase
cbPluginBase::~cbPluginBase
cbPluginBase::GetPaneMask
cbPluginBase::IsReady
cbPluginBase::OnInitPlugin
cbPluginBase::ProcessEvent

---

cbPluginBase
cbpluginbase
browse00279
K cbPluginBase
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `classref')")

$#+K!**cbPluginEvent**

Base class for all control-bar plugin events. This is not a dynamically-creatable class.

Derived from

wxEvent

Include files

<wx/fl/controlbar.h>

Data structures

Members

cbPluginEvent::cbPluginEvent
cbPluginEvent::Clone

---

cbPluginEvent
cbpluginevent
browse00286
K cbPluginEvent
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `classref')")

$#+K!**cbRemoveBarEvent**

Class for bar removal events.

Derived from

cbPluginEvent

Include files

<wx/fl/controlbar.h>

Data structures

Members

cbRemoveBarEvent::cbRemoveBarEvent

---

cbRemoveBarEvent
cbremovebarevent
browse00289
K cbRemoveBarEvent
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `classref')")

$#+K!**cbResizeBarEvent**

Class for bar resize events.

Derived from

cbPluginEvent

Include files

<wx/fl/controlbar.h>

Data structures

Members

cbResizeBarEvent::cbResizeBarEvent

---

cbResizeBarEvent
cbresizebarevent
browse00291
K cbResizeBarEvent
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `classref')")

$#+K!**cbResizeRowEvent**

Class for row resize events.

Derived from

cbPluginEvent

Include files

<wx/fl/controlbar.h>

Data structures

Members

cbResizeRowEvent::cbResizeRowEvent

---

cbResizeRowEvent
cbresizerowevent
browse00293
K cbResizeRowEvent
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `classref')")

$#+K!**cbRightDownEvent**

Class for mouse right down events.

Derived from

cbPluginEvent

Include files

<wx/fl/controlbar.h>

Data structures

Members

cbRightDownEvent::cbRightDownEvent

---

cbRightDownEvent
cbrightdownevent
browse00295
K cbRightDownEvent
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `classref')")

$#+K!**cbRightUpEvent**

Class for mouse right up events.

Derived from

cbPluginEvent

Include files

<wx/fl/controlbar.h>

Data structures

Members

cbRightUpEvent::cbRightUpEvent

---

cbRightUpEvent
cbrightupevent
browse00297
K cbRightUpEvent
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `classref')")

$#+K!**cbRowDragPlugin**

This plugin adds row-dragging functionality to the pane. It handles mouse movement and pane background-erasing plugin events. The behaviour and appearance resembles drag and drop positioning of the toolbar rows in Netscape Communicator 4.xx.

Derived from

cbPluginBase

Include files

<wx/fl/rowdragpl.h>

Data structures

Members

cbRowDragPlugin::cbRowDragPlugin
cbRowDragPlugin::~cbRowDragPlugin
cbRowDragPlugin::CaptureDCArea
cbRowDragPlugin::CheckPrevItemInFocus
cbRowDragPlugin::Clone
cbRowDragPlugin::CollapseRow
cbRowDragPlugin::Draw3DPattern
cbRowDragPlugin::Draw3DRect
cbRowDragPlugin::DrawCollapsedRowIcon
cbRowDragPlugin::DrawCollapsedRowsBorder
cbRowDragPlugin::DrawEmptyRow
cbRowDragPlugin::DrawOrtoRomb
cbRowDragPlugin::DrawRectShade
cbRowDragPlugin::DrawRomb
cbRowDragPlugin::DrawRombShades
cbRowDragPlugin::DrawRowDragHint
cbRowDragPlugin::DrawRowsDragHintsBorder
cbRowDragPlugin::DrawTrianDown
cbRowDragPlugin::DrawTrianRight
cbRowDragPlugin::DrawTrianUp
cbRowDragPlugin::ExpandRow
cbRowDragPlugin::FinishOnScreenDraw
cbRowDragPlugin::GetCollapsedIconsPos
cbRowDragPlugin::GetCollapsedInconRect
cbRowDragPlugin::GetCollapsedRowIconHeight
cbRowDragPlugin::GetFirstRow
cbRowDragPlugin::GetHRowsCountForPane

---

cbRowDragPlugin
browdragplugin
rowse00299
K cbRowDragPlugin
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `classref')")

cbRowDragPlugin::GetRowDragHintWidth
cbRowDragPlugin::GetRowHintRect
cbRowDragPlugin::HitTestCollapsedRowIcon
cbRowDragPlugin::HitTestRowDragHint
cbRowDragPlugin::InsertDraggedRowBefore
cbRowDragPlugin::ItemIsInFocus
cbRowDragPlugin::OnDrawPaneBackground
cbRowDragPlugin::OnInitPlugin
cbRowDragPlugin::OnLButtonDown
cbRowDragPlugin::OnLButtonUp
cbRowDragPlugin::OnMouseMove
cbRowDragPlugin::PrepareForRowDrag
cbRowDragPlugin::SetMouseCapture
cbRowDragPlugin::SetPaneMargins
cbRowDragPlugin::ShowDraggedRow
cbRowDragPlugin::ShowPaneImage
cbRowDragPlugin::UnhighlightItemInFocus

$#+K!**cbRowInfo**

Helper class used internally by the wxFrameLayout class. Holds and manages information about bar rows.

Derived from

wxObject

Include files

<wx/fl/controlbar.h>

Data structures

Members

cbRowInfo::cbRowInfo
cbRowInfo::~cbRowInfo
cbRowInfo::GetFirstBar

---

cbRowInfo
cbrowinfo
browse00344
K cbRowInfo
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `classref')")

$#+K!**cbRowLayoutPlugin**

Simple implementation of a plugin which handles row layout requests sent from a frame layout.

Derived from

cbPluginBase

Include files

<wx/fl/rowlayoutpl.h>

Data structures

Members

cbRowLayoutPlugin::cbRowLayoutPlugin
cbRowLayoutPlugin::AdjustLengthOfInserted
cbRowLayoutPlugin::ApplyLengthRatios
cbRowLayoutPlugin::CalcRowHeight
cbRowLayoutPlugin::CheckIfAtTheBoundary
cbRowLayoutPlugin::DetectBarHandles
cbRowLayoutPlugin::DoInsertBar
cbRowLayoutPlugin::ExpandNotFixedBars
cbRowLayoutPlugin::FitBarsToRange
cbRowLayoutPlugin::GetRowFreeSpace
cbRowLayoutPlugin::InsertBefore
cbRowLayoutPlugin::LayoutItemsVertically
cbRowLayoutPlugin::MinimzeNotFixedBars
cbRowLayoutPlugin::OnInsertBar
cbRowLayoutPlugin::OnLayoutRow
cbRowLayoutPlugin::OnLayoutRows
cbRowLayoutPlugin::OnRemoveBar
cbRowLayoutPlugin::OnResizeRow
cbRowLayoutPlugin::RecalcLengthRatios
cbRowLayoutPlugin::RelayoutNotFixedBarsAround
cbRowLayoutPlugin::ShiftLeftTrashold
cbRowLayoutPlugin::ShiftRightTrashold
cbRowLayoutPlugin::SlideLeftSideBars
cbRowLayoutPlugin::SlideRightSideBars
cbRowLayoutPlugin::StickRightSideBars

---

cbRowLayoutPlugin
cbrowlayoutplugin
browse00348
K cbRowLayoutPlugin
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `classref')")

$#+K!**cbSimpleCustomizationPlugin**

This class enables customization of a bar, popping up a menu and handling basic customization such as floating and horizontal/vertical alignment of the bar.

Derived from

cbPluginBase

Include files

<wx/fl/cbcustom.h>

Data structures

Members

cbSimpleCustomizationPlugin::cbSimpleCustomizationPlugin
cbSimpleCustomizationPlugin::OnCustomizeBar
cbSimpleCustomizationPlugin::OnCustomizeLayout
cbSimpleCustomizationPlugin::OnMenuItemSelected

---

cbSimpleCustomizationPlugin
cbsimplecustomizationplugin
browse00374
K cbSimpleCustomizationPlugin
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `classref')")

$#+K!**cbSimpleUpdatesMgr**

This class implements slightly optimized logic for refreshing the areas of frame layout that actually need to be updated.

Derived from

cbUpdatesManagerBase

Include files

<wx/fl/updatesmgr.h>

Data structures

Members

cbSimpleUpdatesMgr::cbSimpleUpdatesMgr
cbSimpleUpdatesMgr::OnBarWillChange
cbSimpleUpdatesMgr::OnFinishChanges
cbSimpleUpdatesMgr::OnPaneMarginsWillChange
cbSimpleUpdatesMgr::OnPaneWillChange
cbSimpleUpdatesMgr::OnRowWillChange
cbSimpleUpdatesMgr::OnStartChanges
cbSimpleUpdatesMgr::UpdateNow
cbSimpleUpdatesMgr::WasChanged

---

cbSimpleUpdatesMgr
cbsimpleupdatesmgr
browse00379
K cbSimpleUpdatesMgr
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `classref')")

$#+K!**cbSizeBarWndEvent**

Class for bar window resize events.

Derived from

cbPluginEvent

Include files

<wx/fl/controlbar.h>

Data structures

Members

cbSizeBarWndEvent::cbSizeBarWndEvent

---

cbSizeBarWndEvent
cbsizebarwndevent
browse00389
K cbSizeBarWndEvent
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `classref')")

$#+K!cbStartBarDraggingEvent

Class for start-bar-dragging events.

Derived from

cbPluginEvent

Include files

<wx/fl/controlbar.h>

Data structures

Members

cbStartBarDraggingEvent::cbStartBarDraggingEvent

---

cbStartBarDraggingEvent
cbstartbardraggingevent
browse00391
K cbStartBarDraggingEvent
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `classref')")

$#+K!**cbStartDrawInAreaEvent**

Class for start drawing in area events.

Derived from

cbPluginEvent

Include files

<wx/fl/controlbar.h>

Data structures

Members

cbStartDrawInAreaEvent::cbStartDrawInAreaEvent

---

cbStartDrawInAreaEvent
cbstartdrawinareaevent
browse00393
K cbStartDrawInAreaEvent
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `classref')")

$#+K!**cbUpdateMgrData**

A structure that is present in each item of layout, used by any particular updates-manager to store auxiliary information to be used by its updating algorithm.

Derived from

wxObject

Include files

<wx/fl/controlbar.h>

Data structures

Members

cbUpdateMgrData::cbUpdateMgrData
cbUpdateMgrData::IsDirty
cbUpdateMgrData::SetCustomData
cbUpdateMgrData::SetDirty
cbUpdateMgrData::StoreItemState

---

cbUpdateMgrData
cbupdatemgrdata
browse00395
K cbUpdateMgrData
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `classref')")

$#+K!**cbUpdatesManagerBase**

This class declares an abstract interface for optimized logic that should refresh areas of frame layout that actually need to be updated. This should be extended in future to implement a custom updating strategy.

Derived from

wxObject

Include files

<wx/fl/controlbar.h>

Data structures

Members

cbUpdatesManagerBase::cbUpdatesManagerBase
cbUpdatesManagerBase::~cbUpdatesManagerBase
cbUpdatesManagerBase::OnBarWillChange
cbUpdatesManagerBase::OnFinishChanges
cbUpdatesManagerBase::OnPaneMarginsWillChange
cbUpdatesManagerBase::OnPaneWillChange
cbUpdatesManagerBase::OnRowWillChange
cbUpdatesManagerBase::OnStartChanges
cbUpdatesManagerBase::SetLayout
cbUpdatesManagerBase::UpdateNow

---

cbUpdatesManagerBase
cbupdatesmanagerbase
browse00401
K cbUpdatesManagerBase
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `classref')")

$#+K!**wxDynamicToolBar**

wxDynamicToolBar manages containment and layout of tool windows.

Derived from

wxToolBarBase

Include files

<wx/fl/dyntbar.h>

Data structures

Members

wxDynamicToolBar::wxDynamicToolBar
wxDynamicToolBar::~wxDynamicToolBar
wxDynamicToolBar::AddSeparator
wxDynamicToolBar::AddTool
wxDynamicToolBar::Create
wxDynamicToolBar::CreateDefaultLayout
wxDynamicToolBar::CreateTool
wxDynamicToolBar::DoDeleteTool
wxDynamicToolBar::DoEnableTool
wxDynamicToolBar::DoInsertTool
wxDynamicToolBar::DoSetToggle
wxDynamicToolBar::DoToggleTool
wxDynamicToolBar::DrawSeparator
wxDynamicToolBar::EnableTool
wxDynamicToolBar::FindToolForPosition
wxDynamicToolBar::GetPreferredDim
wxDynamicToolBar::GetToolInfo
wxDynamicToolBar::Layout
wxDynamicToolBar::OnEraseBackground
wxDynamicToolBar::OnPaint
wxDynamicToolBar::OnSize
wxDynamicToolBar::Realize
wxDynamicToolBar::RemveTool
wxDynamicToolBar::SetLayout
wxDynamicToolBar::SizeToolWindows

---

wxDynamicToolBar
wxdynamictoolbar
browse00412
K wxDynamicToolBar
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `classref')")

$#+K!**wxDynToolInfo**

This class holds dynamic toolbar item information.

Derived from

wxToolLayoutItem

Include files

<wx/fl/dyntbar.h>

Data structures

---

wxDynToolInfo
wxdyntoolinfo
browse00438
K wxDynToolInfo
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `classref')")

$#+K!**wxFrameLayout**

wxFrameLayout manages containment and docking of control bars, which can be docked along the top, bottom, right, or left side of the parent frame.

Derived from

wxEvtHandler

Include files

<wx/fl/controlbar.h>

Data structures

Members

wxFrameLayout::wxFrameLayout
wxFrameLayout::~wxFrameLayout
wxFrameLayout::Activate
wxFrameLayout::AddBar
wxFrameLayout::AddPlugin
wxFrameLayout::AddPluginBefore
wxFrameLayout::ApplyBarProperties
wxFrameLayout::CanReparent
wxFrameLayout::CaptureEventsForPane
wxFrameLayout::CaptureEventsForPlugin
wxFrameLayout::CreateCursors
wxFrameLayout::CreateUpdatesManager
wxFrameLayout::Deactivate
wxFrameLayout::DestroyBarWindows
wxFrameLayout::DoSetBarState
wxFrameLayout::EnableFloating
wxFrameLayout::FindBarByName
wxFrameLayout::FindBarByWindow
wxFrameLayout::FindPlugin
wxFrameLayout::FirePluginEvent
wxFrameLayout::ForwardMouseEvent
wxFrameLayout::GetBarPane
wxFrameLayout::GetBars
wxFrameLayout::GetClientHeight
wxFrameLayout::GetClientRect
wxFrameLayout::GetClientWidth
wxFrameLayout::GetFrameClient
wxFrameLayout::GetPane

---

wxFrameLayout
wxframelayout
browse00439
wxFrameLayout
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `classref')")

wxFrameLayout::GetPaneProperties
wxFrameLayout::GetPanesArray
wxFrameLayout::GetParentFrame
wxFrameLayout::GetPrevClientRect
wxFrameLayout::GetTopPlugin
wxFrameLayout::GetUpdatesManager
wxFrameLayout::HasTopPlugin
wxFrameLayout::HideBarWindows
wxFrameLayout::HitTestPane
wxFrameLayout::HitTestPanes
wxFrameLayout::HookUpToFrame
wxFrameLayout::InverseVisibility
wxFrameLayout::LocateBar
wxFrameLayout::OnActivate
wxFrameLayout::OnEraseBackground
wxFrameLayout::OnIdle
wxFrameLayout::OnKillFocus
wxFrameLayout::OnLButtonDown
wxFrameLayout::OnLButtonUp
wxFrameLayout::OnLDblClick
wxFrameLayout::OnMouseMove
wxFrameLayout::OnPaint
wxFrameLayout::OnRButtonDown
wxFrameLayout::OnRButtonUp
wxFrameLayout::OnSetFocus
wxFrameLayout::OnSize
wxFrameLayout::PopAllPlugins
wxFrameLayout::PopPlugin
wxFrameLayout::PositionClientWindow
wxFrameLayout::PositionPanes
wxFrameLayout::PushDefaultPlugins
wxFrameLayout::PushPlugin
wxFrameLayout::RecalcLayout
wxFrameLayout::RedockBar
wxFrameLayout::RefreshNow
wxFrameLayout::ReleaseEventsFromPane
wxFrameLayout::ReleaseEventsFromPlugin
wxFrameLayout::RemoveBar
wxFrameLayout::RemovePlugin
wxFrameLayout::ReparentWindow
wxFrameLayout::RepositionFloatedBar
wxFrameLayout::RouteMouseEvent
wxFrameLayout::SetBarState
wxFrameLayout::SetFrameClient
wxFrameLayout::SetMargins
wxFrameLayout::SetPaneBackground
wxFrameLayout::SetPaneProperties
wxFrameLayout::SetTopPlugin
wxFrameLayout::SetUpdatesManager
wxFrameLayout::ShowFloatedWindows
wxFrameLayout::UnhookFromFrame

$#+K!**wxFrameManager**

Derived from

wxObject

Data structures

Members

wxFrameManager::wxFrameManager
wxFrameManager::~wxFrameManager
wxFrameManager::ActivateView
wxFrameManager::AddView
wxFrameManager::DeactivateCurrentView
wxFrameManager::DestroyViews
wxFrameManager::DoSerialize
wxFrameManager::EnableMenusForView
wxFrameManager::GetActiveView
wxFrameManager::GetActiveViewNo
wxFrameManager::GetActiveViewNode
wxFrameManager::GetClientWindow
wxFrameManager::GetObjectStore
wxFrameManager::GetParentFrame
wxFrameManager::GetParentWindow
wxFrameManager::GetView
wxFrameManager::GetViewNo
wxFrameManager::Init
wxFrameManager::ReloadViews
wxFrameManager::RemoveView
wxFrameManager::SaveViewsNow
wxFrameManager::SetClinetWindow
wxFrameManager::SyncAllMenus
wxFrameManager::ViewsAreLoaded

---

wxFrameManager
wxframemanager
browse00519
wxFrameManager
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `classref')")

$#+K!**GarbageCollector**

This class implements an extremely slow but simple garbage collection algorithm.

Derived from

No base class

Include files

<wx/fl/garbagec.h>

Data structures

Members

GarbageCollector::GarbageCollector
GarbageCollector::~GarbageCollector
GarbageCollector::AddDependency
GarbageCollector::AddObject
GarbageCollector::ArrangeCollection
GarbageCollector::DestroyItemList
GarbageCollector::FindItemNode
GarbageCollector::FindReferenceFreeItemNode
GarbageCollector::GetCycledObjects
GarbageCollector::GetRegularObjects
GarbageCollector::RemoveReferencesToNode
GarbageCollector::Reset
GarbageCollector::ResolveReferences

Garbage Collector
garbagecollector
browse00544
K GarbageCollector
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `classref')")

$#+K!**LayoutManagerBase**

This is a base class for layout algorithm implementations.

Derived from

No base class

Include files

<wx/fl/dyntbar.h>

Data structures

Members

LayoutManagerBase::~LayoutManagerBase
LayoutManagerBase::Layout

LayoutManagerBase
layoutmanagerbase
browse00558
LayoutManagerBase
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `classref')")

$#+K!**wxNewBitmapButton**

This is an alternative class to wxBitmapButton. It is used in the implementation of dynamic toolbars.

Derived from

wxPanel

Include files

<wx/fl/newbmpbtn.h>

Data structures

Members

wxNewBitmapButton::wxNewBitmapButton
wxNewBitmapButton::~wxNewBitmapButton
wxNewBitmapButton::DestroyLabels
wxNewBitmapButton::DrawDecorations
wxNewBitmapButton::DrawLabel
wxNewBitmapButton::DrawShade
wxNewBitmapButton::GetStateLabel
wxNewBitmapButton::IsInWindow
wxNewBitmapButton::OnEraseBackground
wxNewBitmapButton::OnKillFocus
wxNewBitmapButton::OnLButtonDown
wxNewBitmapButton::OnLButtonUp
wxNewBitmapButton::OnMouseMove
wxNewBitmapButton::OnPaint
wxNewBitmapButton::OnSize
wxNewBitmapButton::RenderAllLabelImages
wxNewBitmapButton::RenderLabelImage
wxNewBitmapButton::RenderLabelImages
wxNewBitmapButton::Reshape
wxNewBitmapButton::SetAlignments
wxNewBitmapButton::SetLabel

---

wxNewBitmapButton
wxnewbitmapbutton
browse00561
wxNewBitmapButton
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `classref')")

$#+K!**wxToolLayoutItem**

Tool layout item.

Derived from

wxObject

Include files

<wx/fl/dyntbar.h>

Data structures

```
typedef wxToolLayoutItem* wxToolLayoutItemPtrT


typedef wxDynToolInfo* wxDynToolInfoPtrT
```

---

wxToolLayoutItem
wxtoollayoutitem
browse00583
K wxToolLayoutItem
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `classref')")

$#+K!**wxToolWindow**

A tool window is a special kind of frame that paints its own title, and can be used to implement small floating windows.

Derived from

wxFrame

Include files

<wx/fl/toolwnd.h>

Data structures

Members

wxToolWindow::wxToolWindow
wxToolWindow::~wxToolWindow
wxToolWindow::AddMiniButton
wxToolWindow::AdjustRectPos
wxToolWindow::CalcResizedRect
wxToolWindow::DrawHintRect
wxToolWindow::GetClient
wxToolWindow::GetMinimalWndDim
wxToolWindow::GetPreferredSize
wxToolWindow::GetScrMousePos
wxToolWindow::GetScrWindowRect
wxToolWindow::HandleTitleClick
wxToolWindow::HitTestWindow
wxToolWindow::LayoutMiniButtons
wxToolWindow::OnEraseBackground
wxToolWindow::OnLeftDown
wxToolWindow::OnLeftUp
wxToolWindow::OnMiniButtonClicked
wxToolWindow::OnMotion
wxToolWindow::OnPaint
wxToolWindow::OnSize
wxToolWindow::SetClient
wxToolWindow::SetHintCursor
wxToolWindow::SetTitleFont

---

wxToolWindow
wxtoolwindow
browse00584
K wxToolWindow
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `classref')")

$#+K!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
browse00611
K Notes on using the reference
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `overviews')")

$#+K!Event macros and identifiers

These are the event macros and event identifiers defined by FL.

Event macro	Event identifier
EVT_PL_LEFT_DOWN(func)	cbEVT_PL_LEFT_DOWN
EVT_PL_LEFT_UP(func)	cbEVT_PL_LEFT_UP
EVT_PL_RIGHT_DOWN(func)	cbEVT_PL_RIGHT_DOWN
EVT_PL_RIGHT_UP(func)	cbEVT_PL_RIGHT_UP
EVT_PL_MOTION(func)	cbEVT_PL_MOTION
EVT_PL_LEFT_DCLICK(func)	cbEVT_PL_LEFT_DCLICK
EVT_PL_LAYOUT_ROW(func)	cbEVT_PL_LAYOUT_ROW
EVT_PL_RESIZE_ROW(func)	cbEVT_PL_RESIZE_ROW
EVT_PL_LAYOUT_ROWS(func)	cbEVT_PL_LAYOUT_ROWS
EVT_PL_INSERT_BAR(func)	cbEVT_PL_INSERT_BAR
EVT_PL_RESIZE_BAR(func)	cbEVT_PL_RESIZE_BAR
EVT_PL_REMOVE_BAR(func)	cbEVT_PL_REMOVE_BAR
EVT_PL_SIZE_BAR_WND(func)	cbEVT_PL_SIZE_BAR_WND
EVT_PL_DRAW_BAR_DECOR(func)	cbEVT_PL_DRAW_BAR_DECOR
EVT_PL_DRAW_ROW_DECOR(func)	cbEVT_PL_DRAW_ROW_DECOR
EVT_PL_DRAW_PANE_DECOR(func)	cbEVT_PL_DRAW_PANE_DECOR
EVT_PL_DRAW_BAR_HANDLES(func)	cbEVT_PL_DRAW_BAR_HANDLES
EVT_PL_DRAW_ROW_HANDLES(func)	cbEVT_PL_DRAW_ROW_HANDLES
EVT_PL_DRAW_ROW_BKGROUND(func)	cbEVT_PL_DRAW_ROW_BKGROUND
EVT_PL_DRAW_PANE_BKGROUND(func)	cbEVT_PL_DRAW_PANE_BKGROUND
EVT_PL_START_BAR_DRAGGING(func)	cbEVT_PL_START_BAR_DRAGGING

Event macros and identifiers
events
browse00612
K Event macros and identifiers
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `overviews')")

EVT_PL_DRAW_HINT_RECT(func)	cbEVT_PL_DRAW_HINT_RECT
EVT_PL_START_DRAW_IN_AREA(func)	cbEVT_PL_START_DRAW_IN_AREA
EVT_PL_FINISH_DRAW_IN_AREA(func)	cbEVT_PL_FINISH_DRAW_IN_AREA
EVT_PL_CUSTOMIZE_BAR(func)	cbEVT_PL_CUSTOMIZE_BAR
EVT_PL_CUSTOMIZE_LAYOUT(func)	cbEVT_PL_CUSTOMIZE_LAYOUT

See also the classes:

cbCustomizeBarEvent	Class for bar customization events.
cbCustomizeLayoutEvent	Class for layout customization events.
cbDrawBarDecorEvent	Class for bar decoration drawing events.
cbDrawBarHandlesEvent	Class for bar handles drawing events.
cbDrawHintRectEvent	Class for hint-rectangle drawing events.
cbDrawPaneBkGroundEvent	Class for pane background drawing events.
cbDrawPaneDecorEvent	Class for pane decoration drawing events.
cbDrawRowBkGroundEvent	Class for row background drawing events.
cbDrawRowDecorEvent	Class for row decoration drawing events.
cbDrawRowHandlesEvent	Class for row handles drawing events.
cbFinishDrawInAreaEvent	Class for finish drawing in area events.
cbInsertBarEvent	Class for bar insertion events.
cbLayoutRowEvent	Class for single row layout events.
cbLayoutRowsEvent	Class for multiple rows layout events.
cbLeftDClickEvent	Class for mouse left double click events.
cbLeftDownEvent	Class for mouse left down events.
cbLeftUpEvent	Class for mouse left up events.
cbMotionEvent	Class for mouse motion events.
cbPluginEvent	Base class for all control-bar plugin events.
cbRemoveBarEvent	Class for bar removal events.
cbResizeBarEvent	Class for bar resize events.
cbResizeRowEvent	Class for row resize events.
cbRightDownEvent	Class for mouse right down events.

cbRightUpEvent	Class for mouse right up events.
cbSizeBarWndEvent	Class for bar window resize events.
cbStartBarDraggingEvent	Class for start-bar-dragging events.
cbStartDrawInAreaEvent	Class for start drawing in area events.

$#+K!**FAQ**

A row of all non-fixed bars don't position properly

FAQ
faq
browse00613
K FAQ
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `overviews')")

$#+K!**BagLayout::Layout**

void Layout(**const wxSize&** *parentDim*, **wxSize&** *resultingDim*, **wxLayoutItemArrayT&** *items*, **int** *horizGap*, **int** *vertGap*)K

Constructor.

BagLayout::Layout
baglayoutlayout
browse00010
K BagLayout Layout
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `baglayout')")
K Layout

$#+K!**wxBarIterator::wxBarIterator**

wxBarIterator(**RowArrayT&** *rows*)K

Constructor, taking row array.

---

wxBarIterator::wxBarIterator
wxbariteratorwxbariterator
browse00012
K wxBarIterator wxBarIterator
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxbariterator')")
K wxBarIterator

$#+K!**wxBarIterator::BarInfo**

cbBarInfo& BarInfo()[K]

Gets the current bar information.

---

[w]xBarIterator::BarInfo
[w]xbariteratorbarinfo
[b]rowse00013
[K] wxBarIterator BarInfo
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxbariterator')")
[K] BarInfo

$#+K!**wxBarIterator::Next**

bool Next()K

Advances the iterator and returns TRUE if a bar is available.

---

wxBarIterator::Next
wxbariteratornext
browse00014
K wxBarIterator Next
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxbariterator')")
K Next

$#+K!**wxBarIterator::Reset**

void Reset()K

Resets the iterator to the start of the first row.

---

wxBarIterator::Reset
wxbariteratorreset
browse00015
K wxBarIterator Reset
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxbariterator')")
K Reset

$#+K!wxBarIterator::RowInfo

cbRowInfo& RowInfo()[K]

Returns a reference to the currently traversed row.

---

[w]xBarIterator::RowInfo
[w]xbariteratorrowinfo
[b]rowse00016
[K] wxBarIterator RowInfo
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxbariterator')")
[K] RowInfo

$#+K!**cbAntiflickerPlugin::cbAntiflickerPlugin**

cbAntiflickerPlugin()K

Default constructor.

cbAntiflickerPlugin(**wxFrameLayout*** *pPanel*, **int** *paneMask* = *wxALL_PANES*)K

Constructor taking frame layout panel, and pane mask.

---

cbAntiflickerPlugin::cbAntiflickerPlugin
cbantiflickerplugincbantiflickerplugin
browse00018
K cbAntiflickerPlugin cbAntiflickerPlugin
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbantiflickerplugin')")
K cbAntiflickerPlugin
K cbAntiflickerPlugin

$#+K!**cbAntiflickerPlugin::~cbAntiflickerPlugin**

~cbAntiflickerPlugin()K

Destructor.

---

cbAntiflickerPlugin::~cbAntiflickerPlugin
cbantiflickerplugindtor
browse00019
K cbAntiflickerPlugin ~cbAntiflickerPlugin
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbantiflickerplugin')")
K ~cbAntiflickerPlugin

$#+K!**cbAntiflickerPlugin::AllocNewBuffer**

wxDC* AllocNewBuffer(**const wxRect&** *forArea*)K

Allocates a suitable buffer.

---

cbAntiflickerPlugin::AllocNewBuffer
cbantiflickerpluginallocnewbuffer
browse00020
K cbAntiflickerPlugin AllocNewBuffer
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbantiflickerplugin')")
K AllocNewBuffer

$#+K!cbAntiflickerPlugin::FindSuitableBuffer

wxDC* FindSuitableBuffer(**const wxRect&** *forArea*)[K]

Finds a suitable buffer. Returns NULL if a suitable buffer is not present.

---

[c]bAntiflickerPlugin::FindSuitableBuffer
[c]bantiflickerpluginfindsuitablebuffer
[b]rowse00021
[K] cbAntiflickerPlugin FindSuitableBuffer
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbantiflickerplugin')")
[K] FindSuitableBuffer

$#+K!**cbAntiflickerPlugin::GetClientDC**

wxDC& GetClientDC()K

Gets the client device context.

---

cbAntiflickerPlugin::GetClientDC
cbantiflickerplugingetclientdc
browse00022
K cbAntiflickerPlugin GetClientDC
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbantiflickerplugin')")
K GetClientDC

$#+K!**cbAntiflickerPlugin::GetWindowDC**

wxDC& GetWindowDC()K

Gets the window device context.

---

cbAntiflickerPlugin::GetWindowDC
cbantiflickerplugingetwindowdc
browse00023
K cbAntiflickerPlugin GetWindowDC
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbantiflickerplugin')")
K GetWindowDC

$#+K!cbAntiflickerPlugin::OnFinishDrawInArea

void OnFinishDrawInArea(**cbFinishDrawInAreaEvent&** *event*)[K]

Handler for plugin event.

---

[c]bAntiflickerPlugin::OnFinishDrawInArea
[c]bantiflickerpluginonfinishdrawinarea
[b]rowse00024
[K] cbAntiflickerPlugin OnFinishDrawInArea
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbantiflickerplugin')")
[K] OnFinishDrawInArea

$#+K!cbAntiflickerPlugin::OnStartDrawInArea

void OnStartDrawInArea(**cbStartDrawInAreaEvent&** *event*)[K]

Handler for plugin event.

---

[c]bAntiflickerPlugin::OnStartDrawInArea
[c]bantiflickerpluginonstartdrawinarea
[b]rowse00025
[K] cbAntiflickerPlugin OnStartDrawInArea
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbantiflickerplugin')")
[K] OnStartDrawInArea

$#+K!cbBarDimHandlerBase::cbBarDimHandlerBase

cbBarDimHandlerBase()K

to multiple bars, it's instance is reference-counted Default constructor. The initial reference count is 0, since the handler is not used until the first invocation of AddRef().

---

cbBarDimHandlerBase::cbBarDimHandlerBase
cbbardimhandlerbasecbbardimhandlerbase
browse00027
K cbBarDimHandlerBase cbBarDimHandlerBase
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbbardimhandlerbase')")
K cbBarDimHandlerBase

$#+K!**cbBarDimHandlerBase::AddRef**

void AddRef()K

Increments the reference count.

---

cbBarDimHandlerBase::AddRef
cbbardimhandlerbaseaddref
browse00028
K cbBarDimHandlerBase AddRef
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbbardimhandlerbase')")
K AddRef

$#+K!**cbBarDimHandlerBase::OnChangeBarState**

void OnChangeBarState(**cbBarInfo*** *pBar*, **int** *newState*)K

Responds to "bar-state-changes" notifications.

---

cbBarDimHandlerBase::OnChangeBarState
cbbardimhandlerbaseonchangebarstate
browse00029
K cbBarDimHandlerBase OnChangeBarState
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbbardimhandlerbase')")
K OnChangeBarState

$#+K!**cbBarDimHandlerBase::OnResizeBar**

void OnResizeBar(**cbBarInfo*** *pBar*, **const wxSize&** *given*, **wxSize&** *preferred*)K

Responds to bar resize notifications.

---

cbBarDimHandlerBase::OnResizeBar
cbbardimhandlerbaseonresizebar
browse00030
K cbBarDimHandlerBase OnResizeBar
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbbardimhandlerbase')")
K OnResizeBar

$#+K!**cbBarDimHandlerBase::RemoveRef**

void RemoveRef()K

Decrements the reference count, and if the count is at zero, delete 'this'.

---

cbBarDimHandlerBase::RemoveRef
cbbardimhandlerbaseremoveref
browse00031
K cbBarDimHandlerBase RemoveRef
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbbardimhandlerbase')")
K RemoveRef

$#+K!cbBarDragPlugin::cbBarDragPlugin

cbBarDragPlugin()K

Default constructor.

cbBarDragPlugin(**wxFrameLayout*** *pPanel*, **int** *paneMask* = *wxALL_PANES*)K

Constructor taking a parent frame, and flag. See cbPluginBase.

---

cbBarDragPlugin::cbBarDragPlugin
cbbardragplugincbbardragplugin
browse00033
K cbBarDragPlugin cbBarDragPlugin
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbbardragplugin')")
K cbBarDragPlugin
K cbBarDragPlugin

$#+K!**cbBarDragPlugin::~cbBarDragPlugin**

~cbBarDragPlugin()K

Destructor.

---

cbBarDragPlugin::~cbBarDragPlugin
cbbardragplugindtor
browse00034
K cbBarDragPlugin ~cbBarDragPlugin
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbbardragplugin')")
K ~cbBarDragPlugin

$#+K!**cbBarDragPlugin::AdjustHintRect**

void AdjustHintRect(**wxPoint&** *mousePos*)K

the thicker rectangle is drawn using hatched brush, the default border width for this rectangle is 8 pix. Internal implementation function.

---

cbBarDragPlugin::AdjustHintRect
cbbardragpluginadjusthintrect
browse00035
K cbBarDragPlugin AdjustHintRect
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbbardragplugin')")
K AdjustHintRect

$#+K!cbBarDragPlugin::CalcOnScreenDims

void CalcOnScreenDims(**wxRect&** *rect*)K

Internal implementation function.

---

cbBarDragPlugin::CalcOnScreenDims
cbbardragplugincalconscreendims
browse00036
K cbBarDragPlugin CalcOnScreenDims
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbbardragplugin')")
K CalcOnScreenDims

$#+K!cbBarDragPlugin::ClipPosInFrame

void ClipPosInFrame(**wxPoint&** *pos*)K

Internal implementation function.

---

cbBarDragPlugin::ClipPosInFrame
cbbardragpluginclipposinframe
browse00037
K cbBarDragPlugin ClipPosInFrame
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbbardragplugin')")
K ClipPosInFrame

$#+K!**cbBarDragPlugin::ClipRectInFrame**

void ClipRectInFrame(**wxRect&** *rect*)K

Internal implementation function.

---

cbBarDragPlugin::ClipRectInFrame
cbbardragplugincliprectinframe
browse00038
K cbBarDragPlugin ClipRectInFrame
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbbardragplugin')")
K ClipRectInFrame

$#+K!**cbBarDragPlugin::DoDrawHintRect**

void DoDrawHintRect(**wxRect&** *rect*, **bool** *isInClientRect*)[K]

Internal implementation function. Draw the hint rectangle.

---

[c]bBarDragPlugin::DoDrawHintRect
[c]bbardragplugindodrawhintrect
[b]rowse00039
[K] cbBarDragPlugin DoDrawHintRect
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbbardragplugin')")
[K] DoDrawHintRect

$#+K!**cbBarDragPlugin::DrawHintRect**

void DrawHintRect(**wxRect&** *rect*, **bool** *isInClientRect*)K

Internal implementation function. Draw the visual hint while dragging.

---

cbBarDragPlugin::DrawHintRect
cbbardragplugindrawhintrect
browse00040
K cbBarDragPlugin DrawHintRect
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbbardragplugin')")
K DrawHintRect

$#+K!**cbBarDragPlugin::EraseHintRect**

void EraseHintRect(**wxRect&** *rect*, **bool** *isInClientRect*)K

Internal implementation function. Erase the visual hint while dragging.

---

cbBarDragPlugin::EraseHintRect
cbbardragpluginerasehintrect
browse00041
K cbBarDragPlugin EraseHintRect
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbbardragplugin')")
K EraseHintRect

$#+K!**cbBarDragPlugin::FinishTracking**

void FinishTracking()K

Internal implementation function. Stop showing the visual hint while dragging.

---

cbBarDragPlugin::FinishTracking
cbbardragpluginfinishtracking
browse00042
K cbBarDragPlugin FinishTracking
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbbardragplugin')")
K FinishTracking

$#+K!cbBarDragPlugin::GetBarHeightInPane

int GetBarHeightInPane(**cbDockPane*** *pPane*)K

Internal implementation function.

---

cbBarDragPlugin::GetBarHeightInPane
cbbardragplugingetbarheightinpane
browse00043
K cbBarDragPlugin GetBarHeightInPane
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbbardragplugin')")
K GetBarHeightInPane

$#+K!cbBarDragPlugin::GetBarWidthInPane

int GetBarWidthInPane(**cbDockPane*** *pPane*)K

Internal implementation function.

---

cbBarDragPlugin::GetBarWidthInPane
cbbardragplugingetbarwidthinpane
browse00044
K cbBarDragPlugin GetBarWidthInPane
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbbardragplugin')")
K GetBarWidthInPane

$#+K!cbBarDragPlugin::GetDistanceToPane

int GetDistanceToPane(**cbDockPane*** *pPane*, **wxPoint&** *mousePos*)[K]

Internal implementation function.

---

[c]bBarDragPlugin::GetDistanceToPane
[c]bbardragplugingetdistancetopane
[b]rowse00045
[K] cbBarDragPlugin GetDistanceToPane
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbbardragplugin')")
[K] GetDistanceToPane

$#+K!cbBarDragPlugin::HitTestPanes

cbDockPane* HitTestPanes(**wxRect&** *rect*)K

Internal implementation function. Finds the pane under the specified rectangle.

cbDockPane* HitTestPanes(**wxPoint&** *pos*)K

Internal implementation function. Finds the pane under the specified point.

---

cbBarDragPlugin::HitTestPanes
cbbardragpluginhittestpanes
browse00046
K cbBarDragPlugin HitTestPanes
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbbardragplugin')")
K HitTestPanes
K HitTestPanes

$#+K!**cbBarDragPlugin::HitsPane**

bool HitsPane(**cbDockPane*** *pPane*, **wxRect&** *rect*)K

Internal implementation function.

---

cbBarDragPlugin::HitsPane
cbbardragpluginhitspane
browse00047
K cbBarDragPlugin HitsPane
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbbardragplugin')")
K HitsPane

$#+K!**cbBarDragPlugin::IsInClientArea**

bool IsInClientArea(**wxPoint&** *mousePos*)K

Internal implementation function.

bool IsInClientArea(**wxRect&** *rect*)K

Internal implementation function.

---

cbBarDragPlugin::IsInClientArea
cbbardragpluginisinclientarea
browse00048
K cbBarDragPlugin IsInClientArea
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbbardragplugin')")
K IsInClientArea
K IsInClientArea

$#+K!cbBarDragPlugin::IsInOtherPane

bool IsInOtherPane(**wxPoint&** *mousePos*)K

Internal implementation function.

---

cbBarDragPlugin::IsInOtherPane
cbbardragpluginisinotherpane
browse00049
K cbBarDragPlugin IsInOtherPane
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbbardragplugin')")
K IsInOtherPane

$#+K!**cbBarDragPlugin::OnDrawHintRect**

void OnDrawHintRect(**cbDrawHintRectEvent&** *event*)K

Handles event, which originates from itself.

---

cbBarDragPlugin::OnDrawHintRect
cbbardragpluginondrawhintrect
browse00050
K cbBarDragPlugin OnDrawHintRect
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbbardragplugin')")
K OnDrawHintRect

$#+K!**cbBarDragPlugin::OnLButtonDown**

void OnLButtonDown(**cbLeftDownEvent&** *event*)K

Handler for plugin event.

---

cbBarDragPlugin::OnLButtonDown
cbbardragpluginonlbuttondown
browse00051
K cbBarDragPlugin OnLButtonDown
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbbardragplugin')")
K OnLButtonDown

$#+K!**cbBarDragPlugin::OnLButtonUp**

void OnLButtonUp(**cbLeftUpEvent&** *event*)K

Handler for plugin event.

---

cbBarDragPlugin::OnLButtonUp
cbbardragpluginonlbuttonup
browse00052
K cbBarDragPlugin OnLButtonUp
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbbardragplugin')")
K OnLButtonUp

$#+K!**cbBarDragPlugin::OnLDblClick**

void OnLDblClick(**cbLeftDClickEvent&** *event*)K

Handler for plugin event.

---

cbBarDragPlugin::OnLDblClick
cbbardragpluginonldblclick
browse00053
K cbBarDragPlugin OnLDblClick
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbbardragplugin')")
K OnLDblClick

$#+K!cbBarDragPlugin::OnMouseMove

void OnMouseMove(**cbMotionEvent&** *event*)K

Handler for plugin event.

---

cbBarDragPlugin::OnMouseMove
cbbardragpluginonmousemove
browse00054
K cbBarDragPlugin OnMouseMove
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbbardragplugin')")
K OnMouseMove

$#+K!cbBarDragPlugin::OnStartBarDragging

void OnStartBarDragging(**cbStartBarDraggingEvent&** *event*)K

Handler for plugin event.

---

cbBarDragPlugin::OnStartBarDragging
cbbardragpluginonstartbardragging
browse00055
K cbBarDragPlugin OnStartBarDragging
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbbardragplugin')")
K OnStartBarDragging

$#+K!**cbBarDragPlugin::RectToScr**

void RectToScr(**wxRect&** *frameRect*, **wxRect&** *scrRect*)K

Internal implementation function. Converts the given rectangle from window to screen coordinates.

---

cbBarDragPlugin::RectToScr
cbbardragpluginrecttoscr
browse00056
K cbBarDragPlugin RectToScr
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbbardragplugin')")
K RectToScr

$#+K!**cbBarDragPlugin::ShowHint**

void ShowHint(**bool** *prevWasInClient*)K

Internal implementation function. Show the hint; called within OnMouseMove.

---

cbBarDragPlugin::ShowHint
cbbardragpluginshowhint
browse00057
K cbBarDragPlugin ShowHint
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbbardragplugin')")
K ShowHint

$#+K!**cbBarDragPlugin::StartTracking**

void StartTracking()K

on-screen hint-tracking related methods Internal implementation function. Start showing a visual hint while dragging.

---

cbBarDragPlugin::StartTracking
cbbardragpluginstarttracking
browse00058
K cbBarDragPlugin StartTracking
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbbardragplugin')")
K StartTracking

$#+K! cbBarDragPlugin::StickToPane

void StickToPane(**cbDockPane*** *pPane*, **wxPoint&** *mousePos*)K

Internal implementation function.

---

cbBarDragPlugin::StickToPane
cbbardragpluginsticktopane
browse00059
K cbBarDragPlugin StickToPane
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbbardragplugin')")
K StickToPane

$#+K!cbBarDragPlugin::UnstickFromPane

void UnstickFromPane(**cbDockPane*** *pPane*, **wxPoint&** *mousePos*)[K]

Internal implementation function.

---

[c]bBarDragPlugin::UnstickFromPane
[c]bbardragpluginunstickfrompane
[b]rowse00060
[K] cbBarDragPlugin UnstickFromPane
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbbardragplugin')")
[K] UnstickFromPane

$#+K!cbBarHintsPlugin::cbBarHintsPlugin

cbBarHintsPlugin()K

Default constructor.

cbBarHintsPlugin(**wxFrameLayout*** *pLayout*, **int** *paneMask = wxALL_PANES*)K

Constructor, taking parent frame and pane mask flag.

---

cbBarHintsPlugin::cbBarHintsPlugin
cbbarhintsplugincbbarhintsplugin
browse00062
K cbBarHintsPlugin cbBarHintsPlugin
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbbarhintsplugin')")
K cbBarHintsPlugin
K cbBarHintsPlugin

$#+K!**cbBarHintsPlugin::~cbBarHintsPlugin**

~cbBarHintsPlugin()K

Destructor.

---

cbBarHintsPlugin::~cbBarHintsPlugin
cbbarhintsplugindtor
browse00063
K cbBarHintsPlugin ~cbBarHintsPlugin
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbbarhintsplugin')")
K ~cbBarHintsPlugin

$#+K!**cbBarHintsPlugin::CreateBoxes**

void CreateBoxes()K

Helper function: creates close and collapse boxes.

---

cbBarHintsPlugin::CreateBoxes
cbbarhintsplugincreateboxes
browse00064
K cbBarHintsPlugin CreateBoxes
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbbarhintsplugin')")
K CreateBoxes

$#+K!**cbBarHintsPlugin::DoDrawHint**

void DoDrawHint(**wxDC&** *dc*, **wxRect&** *rect*, **int** *pos*, **int** *boxOfs*, **int** *grooveOfs*, **bool** *isFixed*)K

Helper function: draws a hint.

---

cbBarHintsPlugin::DoDrawHint
cbbarhintsplugindodrawhint
browse00065
K cbBarHintsPlugin DoDrawHint
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbbarhintsplugin')")
K DoDrawHint

$#+K!cbBarHintsPlugin::Draw3DBox

void Draw3DBox(**wxDC&** *dc*, **const wxPoint&** *pos*, **bool** *pressed*)[K]

Helper function: draws a 3D box.

---

[c]bBarHintsPlugin::Draw3DBox
[c]bbarhintsplugindraw3dbox
[b]rowse00066
[K] cbBarHintsPlugin Draw3DBox
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbbarhintsplugin')")
[K] Draw3DBox

$#+K!**cbBarHintsPlugin::DrawCloseBox**

void DrawCloseBox(**wxDC&** *dc*, **const wxPoint&** *pos*, **bool** *pressed*)K

Helper function: draws a close box.

---

cbBarHintsPlugin::DrawCloseBox
cbbarhintsplugindrawclosebox
browse00067
K cbBarHintsPlugin DrawCloseBox
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbbarhintsplugin')")
K DrawCloseBox

$#+K!**cbBarHintsPlugin::DrawCollapseBox**

void DrawCollapseBox(**wxDC&** *dc*, **const wxPoint&** *pos*, **bool** *atLeft*, **bool** *disabled*, **bool** *pressed*)K

Helper function: draws a collapse box.

---

cbBarHintsPlugin::DrawCollapseBox
cbbarhintsplugindrawcollapsebox
browse00068
K cbBarHintsPlugin DrawCollapseBox
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbbarhintsplugin')")
K DrawCollapseBox

$#+K!**cbBarHintsPlugin::DrawGrooves**

void DrawGrooves(**wxDC&** *dc*, **const wxPoint&** *pos*, **int** *length*)K

Helper function: draws grooves.

---

cbBarHintsPlugin::DrawGrooves
cbbarhintsplugindrawgrooves
browse00069
K cbBarHintsPlugin DrawGrooves
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbbarhintsplugin')")
K DrawGrooves

$#+K!**cbBarHintsPlugin::ExcludeHints**

void ExcludeHints(**wxRect&** *rect*, **cbBarInfo&** *info*)K

Helper function.

---

cbBarHintsPlugin::ExcludeHints
cbbarhintspluginexcludehints
browse00070
K cbBarHintsPlugin ExcludeHints
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbbarhintsplugin')")
K ExcludeHints

$#+K!**cbBarHintsPlugin::GetHintsLayout**

void GetHintsLayout(**wxRect&** *rect*, **cbBarInfo&** *info*, **int&** *boxOfs*, **int&** *grooveOfs*, **int&** *pos*)K

Helper function: gets the layout of a hint.

---

cbBarHintsPlugin::GetHintsLayout
cbbarhintsplugingethintslayout
browse00071
K cbBarHintsPlugin GetHintsLayout
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbbarhintsplugin')")
K GetHintsLayout

$#+K!cbBarHintsPlugin::HitTestHints

int HitTestHints(**cbBarInfo&** *info*, **const wxPoint&** *pos*)[K]

Helper function: returns information about the hint under the given position.

---

[c]bBarHintsPlugin::HitTestHints
[c]bbarhintspluginhittesthints
[b]rowse00072
[K] cbBarHintsPlugin HitTestHints
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbbarhintsplugin')")
[K] HitTestHints

$#+K!cbBarHintsPlugin::OnDrawBarDecorations

void OnDrawBarDecorations(**cbDrawBarDecorEvent&** *event*)K

Handles a plugin event.

---

cbBarHintsPlugin::OnDrawBarDecorations
cbbarhintspluginondrawbardecorations
browse00073
K cbBarHintsPlugin OnDrawBarDecorations
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbbarhintsplugin')")
K OnDrawBarDecorations

$#+K!cbBarHintsPlugin::OnInitPlugin

void OnInitPlugin()K

Called to initialize this plugin.

---

cbBarHintsPlugin::OnInitPlugin
cbbarhintspluginoninitplugin
browse00074
K cbBarHintsPlugin OnInitPlugin
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbbarhintsplugin')")
K OnInitPlugin

$#+K!cbBarHintsPlugin::OnLeftDown

void OnLeftDown(cbLeftDownEvent& *event*)K

Handles a plugin event.

---

cbBarHintsPlugin::OnLeftDown
cbbarhintspluginonleftdown
browse00075
K cbBarHintsPlugin OnLeftDown
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbbarhintsplugin')")
K OnLeftDown

$#+K!**cbBarHintsPlugin::OnLeftUp**

void OnLeftUp(**cbLeftUpEvent&** *event*)K

Handles a plugin event.

---

cbBarHintsPlugin::OnLeftUp
cbbarhintspluginonleftup
browse00076
K cbBarHintsPlugin OnLeftUp
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbbarhintsplugin')")
K OnLeftUp

$#+K!cbBarHintsPlugin::OnMotion

void OnMotion(**cbMotionEvent&** *event*)[K]

Handles a plugin event.

---

[c]bBarHintsPlugin::OnMotion
[c]bbarhintspluginonmotion
[b]rowse00077
[K] cbBarHintsPlugin OnMotion
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbbarhintsplugin')")
[K] OnMotion

$#+K!cbBarHintsPlugin::OnSizeBarWindow

void OnSizeBarWindow(**cbSizeBarWndEvent&** *event*)K

Handles a plugin event.

---

cbBarHintsPlugin::OnSizeBarWindow
cbbarhintspluginonsizebarwindow
browse00078
K cbBarHintsPlugin OnSizeBarWindow
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbbarhintsplugin')")
K OnSizeBarWindow

$#+K!cbBarHintsPlugin::SetGrooveCount

void SetGrooveCount(**int** *nGrooves*)K

Set the number of grooves to be shown in the pane.

---

cbBarHintsPlugin::SetGrooveCount
cbbarhintspluginsetgroovecount
browse00079
K cbBarHintsPlugin SetGrooveCount
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbbarhintsplugin')")
K SetGrooveCount

$#+K!**cbBarInfo::cbBarInfo**

cbBarInfo()[K]

Constructor.

---

[c]bBarInfo::cbBarInfo
[c]bbarinfocbbarinfo
[b]rowse00081
[K] cbBarInfo cbBarInfo
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbbarinfo')")
[K] cbBarInfo

$#+K!**cbBarInfo::~cbBarInfo**

~cbBarInfo()K

Destructor.

---

cbBarInfo::~cbBarInfo
cbbarinfodtor
browse00082
K cbBarInfo ~cbBarInfo
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbbarinfo')")
K ~cbBarInfo

$#+KK!**cbBarInfo::IsExpanded**

bool IsExpanded() **const**

Returns TRUE if this bar is expanded.

---

cbBarInfo::IsExpanded
cbbarinfoisexpanded
browse00083
K cbBarInfo IsExpanded
K IsExpanded
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbbarinfo')")

$#+KK!**cbBarInfo::IsFixed**

bool IsFixed() **const**

Returns TRUE if this bar is fixed.

---

[c]bBarInfo::IsFixed
[c]bbarinfoisfixed
[b]rowse00084
[K] cbBarInfo IsFixed
[K] IsFixed
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbbarinfo')")

$#+K!**cbBarSpy::cbBarSpy**

cbBarSpy(**wxFrameLayout*** *pPanel*)K

Constructor, taking a parent pane.

cbBarSpy()K

Default constructor.

---

cbBarSpy::cbBarSpy
cbbarspycbbarspy
browse00087
K cbBarSpy cbBarSpy
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbbarspy')")
K cbBarSpy
K cbBarSpy

$#+K!**cbBarSpy::ProcessEvent**

bool ProcessEvent(**wxEvent&** *event*)K

Performs special event processing.

---

cbBarSpy::ProcessEvent
cbbarspyprocessevent
browse00088
K cbBarSpy ProcessEvent
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbbarspy')")
K ProcessEvent

$#+K!**cbBarSpy::SetBarWindow**

void SetBarWindow(**wxWindow*** *pWnd*)K

Sets the bar window.

---

cbBarSpy::SetBarWindow
cbbarspysetbarwindow
browse00089
K cbBarSpy SetBarWindow
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbbarspy')")
K SetBarWindow

$#+K!**cbCloseBox::Draw**

void Draw(**wxDC&** *dc*)K

Draws the close button appearance.

---

cbCloseBox::Draw
cbcloseboxdraw
browse00091
K cbCloseBox Draw
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbclosebox')")
K Draw

$#+K!**cbCollapseBox::Draw**

void Draw(**wxDC&** *dc*)K

Draws the collapse button appearance.

---

cbCollapseBox::Draw
cbcollapseboxdraw
browse00093
K cbCollapseBox Draw
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbcollapsebox')")
K Draw

$#+K!**cbCustomizeBarEvent::cbCustomizeBarEvent**

cbCustomizeBarEvent(**cbBarInfo*** *pBar*, **const wxPoint&** *clickPos*, **cbDockPane*** *pPane*)K

Constructor, taking bar information, mouse position, and pane.

---

cbCustomizeBarEvent::cbCustomizeBarEvent
cbcustomizebareventcbcustomizebarevent
browse00096
K cbCustomizeBarEvent cbCustomizeBarEvent
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbcustomizebarevent')")
K cbCustomizeBarEvent

$#+K!cbCustomizeLayoutEvent::cbCustomizeLayoutEvent

cbCustomizeLayoutEvent(**const wxPoint&** *clickPos*)K

Constructor, taking mouse position.

---

cbCustomizeLayoutEvent::cbCustomizeLayoutEvent
cbcustomizelayouteventcbcustomizelayoutevent
browse00098
K cbCustomizeLayoutEvent cbCustomizeLayoutEvent
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbcustomizelayoutevent')")
K cbCustomizeLayoutEvent

$#+K!**cbDimInfo::cbDimInfo**

cbDimInfo(**cbBarDimHandlerBase*** *pDimHandler*, **bool** *isFixed*)[K]

Constructor. isFixed is TRUE if vertical/horizontal dimensions cannot be manually adjusted by the user using resizing handles. If FALSE, the frame-layout automatically places resizing handles among bars that do are not fixed.

cbDimInfo(**int** *dh_x*, **int** *dh_y*, **int** *dv_x*, **int** *dv_y*, **int** *f_x*, **int** *f_y*, **bool** *isFixed = TRUE*, **int** *horizGap = 6*, **int** *vertGap = 6*, **cbBarDimHandlerBase*** *pDimHandler = NULL*)[K]

Constructor taking dimenstion information. dh_x, dh_y are the dimensions when docked horizontally. dv_x, dv_y are the dimensions when docked vertically. f_x, f_y are the dimensions when floating. For information on isFixed, see comments above. horizGap is the left/right gap, separating decorations from the bar's actual wndow, filled with the frame's background colour. The dimension is given in the frame's coordinates. vertGap is the top/bottom gap, separating decorations from the bar's actual wndow, filled with the frame's background colour. The dimension is given in the frame's coordinates.

cbDimInfo(**int** *x*, **int** *y*, **bool** *isFixed = TRUE*, **int** *gap = 6*, **cbBarDimHandlerBase*** *pDimHandler = NULL*)[K]

Constructor.

cbDimInfo()[K]

Default constructor.

---

[c]bDimInfo::cbDimInfo
[c]bdiminfocbdiminfo
[b]rowse00100
[K] cbDimInfo cbDimInfo
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbdiminfo')")
[K] cbDimInfo
[K] cbDimInfo
[K] cbDimInfo
[K] cbDimInfo

$#+K!**cbDimInfo::~cbDimInfo**

~cbDimInfo()[K]

Destructor. Destroys handler automatically, if present.

---

[c]bDimInfo::~cbDimInfo
[c]bdiminfodtor
[b]rowse00101
[K] cbDimInfo ~cbDimInfo
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbdiminfo')")
[K] ~cbDimInfo

$#+K!**cbDimInfo::GetDimHandler**

cbBarDimHandlerBase* GetDimHandler()K

Returns the handler, if any.

---

cbDimInfo::GetDimHandler
cbdiminfogetdimhandler
browse00102
K cbDimInfo GetDimHandler
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbdiminfo')")
K GetDimHandler

$#+K!**cbDimInfo::operator=**

const cbDimInfo& operator operator=(**const cbDimInfo&** *other*)[K]

Assignment operator.

---

[c]bDimInfo::operator=
[c]bdiminfooperatorassign
[b]rowse00103
[K] cbDimInfo operator=
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbdiminfo')")
[K] operator=

$#+K!**cbDockBox::Draw**

void Draw(**wxDC&** *dc*)K

Draws the dock button appearance.

---

cbDockBox::Draw
cbdockboxdraw
browse00105
K cbDockBox Draw
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbdockbox')")
K Draw

$#+K!**cbDockPane::cbDockPane**

cbDockPane(**int** *alignment*, **wxFrameLayout*** *pPanel*)K

Constructor, taking alignment and layout panel.

cbDockPane()K

public members Default constructor.

---

cbDockPane::cbDockPane
cbdockpanecbdockpane
browse00107
K cbDockPane cbDockPane
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbdockpane')")
K cbDockPane
K cbDockPane

$#+K!**cbDockPane::~cbDockPane**

~cbDockPane()K

Destructor.

---

cbDockPane::~cbDockPane
cbdockpanedtor
browse00108
K cbDockPane ~cbDockPane
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbdockpane')")
K ~cbDockPane

$#+K!cbDockPane::BarPresent

bool BarPresent(**cbBarInfo*** *pBar*)K

Returns TRUE if the given bar is present in this pane.

---

cbDockPane::BarPresent
cbdockpanebarpresent
browse00109
K cbDockPane BarPresent
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbdockpane')")
K BarPresent

$#+K!cbDockPane::CalcLengthRatios

void CalcLengthRatios(**cbRowInfo*** *pInRow*)K

Calculate lengths. Internal function called by plugins.

---

cbDockPane::CalcLengthRatios
cbdockpanecalclengthratios
browse00110
K cbDockPane CalcLengthRatios
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbdockpane')")
K CalcLengthRatios

$#+K!cbDockPane::ContractBar

void ContractBar(**cbBarInfo*** *pBar*)[K]

Contracts the bar. Internal function called by plugins.

---

[c]bDockPane::ContractBar
[c]bdockpanecontractbar
[b]rowse00111
[K] cbDockPane ContractBar
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbdockpane')")
[K] ContractBar

$#+K!cbDockPane::DoInsertBar

void DoInsertBar(**cbBarInfo*** *pBar*, **int** *rowNo*)[K]

Inserts the bar at the given row number. Internal function called by plugins.

---

[c]bDockPane::DoInsertBar
[c]bdockpanedoinsertbar
[b]rowse00112
[K] cbDockPane DoInsertBar
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbdockpane')")
[K] DoInsertBar

$#+K!cbDockPane::DrawHorizHandle

void DrawHorizHandle(**wxDC&** *dc*, **int** *x*, **int** *y*, **int** *width*)K

Row/bar resizing related helper-method.

---

cbDockPane::DrawHorizHandle
cbdockpanedrawhorizhandle
browse00113
K cbDockPane DrawHorizHandle
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbdockpane')")
K DrawHorizHandle

$#+K!cbDockPane::DrawVertHandle

void DrawVertHandle(**wxDC&** *dc*, **int** *x*, **int** *y*, **int** *height*)[K]

protected really (accessed only by plugins) Row/bar resizing related helper-method.

---

[c]bDockPane::DrawVertHandle
[c]bdockpanedrawverthandle
[b]rowse00114
[K] cbDockPane DrawVertHandle
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbdockpane')")
[K] DrawVertHandle

$#+K!cbDockPane::ExpandBar

void ExpandBar(**cbBarInfo*** *pBar*)K

Expands the bar. Internal function called by plugins.

---

cbDockPane::ExpandBar
cbdockpaneexpandbar
browse00115
K cbDockPane ExpandBar
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbdockpane')")
K ExpandBar

$#+K! cbDockPane::FinishDrawInArea

void FinishDrawInArea(**const wxRect&** *area*)[K]

Generates cbFinishDrawInAreaEvent and sends it to the layout. Internal function called by plugins.

---

[c]bDockPane::FinishDrawInArea
[c]bdockpanefinishdrawinarea
[b]rowse00116
[K] cbDockPane FinishDrawInArea
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbdockpane')")
[K] FinishDrawInArea

$#+K!**cbDockPane::FrameToPane**

void FrameToPane(**int*** *x*, **int*** *y*)K

Coordinate translation between parent's frame and this pane. Internal function called by plugins.

void FrameToPane(**wxRect*** *pRect*)K

Coordinate translation between parent's frame and this pane. Internal function called by plugins.

---

cbDockPane::FrameToPane
cbdockpaneframetopane
browse00117
K cbDockPane FrameToPane
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbdockpane')")
K FrameToPane
K FrameToPane

$#+K!cbDockPane::GetAlignment

int GetAlignment()K

Returns the alignment for this pane. The value is one of FL_ALIGN_TOP, FL_ALIGN_BOTTOM, FL_ALIGN_LEFT, FL_ALIGN_RIGHT.

---

cbDockPane::GetAlignment
cbdockpanegetalignment
browse00118
K cbDockPane GetAlignment
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbdockpane')")
K GetAlignment

$#+K!cbDockPane::GetBarInfoByWindow

cbBarInfo* GetBarInfoByWindow(**wxWindow*** *pBarWnd*)[K]

Finds the bar information by corresponding window.

---

[c]bDockPane::GetBarInfoByWindow
[c]bdockpanegetbarinfobywindow
[b]rowse00119
[K] cbDockPane GetBarInfoByWindow
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbdockpane')")
[K] GetBarInfoByWindow

$#+K!cbDockPane::GetBarResizeRange

void GetBarResizeRange(**cbBarInfo*** *pBar*, **int*** *from*, **int*** *till*, **bool** *forLeftHandle*)K

Returns the bar's resize range.

---

cbDockPane::GetBarResizeRange
cbdockpanegetbarresizerange
browse00120
K cbDockPane GetBarResizeRange
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbdockpane')")
K GetBarResizeRange

$#+K!cbDockPane::GetDockingState

int GetDockingState()K

Returns wxCBAR_DOCKED_HORIZONTALLY if the alignment is top or bottom, or wxCBAR_DOCKED_VERTICALLY otherwise.

---

cbDockPane::GetDockingState
cbdockpanegetdockingstate
browse00121
K cbDockPane GetDockingState
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbdockpane')")
K GetDockingState

$#+K!**cbDockPane::GetFirstRow**

cbRowInfo* GetFirstRow()K

Returns the first row.

---

cbDockPane::GetFirstRow
cbdockpanegetfirstrow
browse00122
K cbDockPane GetFirstRow
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbdockpane')")
K GetFirstRow

$#+K!cbDockPane::GetMinimalRowHeight

int GetMinimalRowHeight(cbRowInfo* *pRow*)[K]

Returns the minimal row height for the given row. Internal function called by plugins.

---

[c]bDockPane::GetMinimalRowHeight
[c]bdockpanegetminimalrowheight
[b]rowse00123
[K] cbDockPane GetMinimalRowHeight
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbdockpane')")
[K] GetMinimalRowHeight

$#+K!cbDockPane::GetNotFixedBarsCount

int GetNotFixedBarsCount(**cbRowInfo*** *pRow*)K

Returns the number of bars whose size is not fixed. Internal function called by plugins.

---

cbDockPane::GetNotFixedBarsCount
cbdockpanegetnotfixedbarscount
browse00124
K cbDockPane GetNotFixedBarsCount
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbdockpane')")
K GetNotFixedBarsCount

$#+K!cbDockPane::GetPaneHeight

int GetPaneHeight()K

Returns the height in the pane's coordinates.

---

cbDockPane::GetPaneHeight
cbdockpanegetpaneheight
browse00125
K cbDockPane GetPaneHeight
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbdockpane')")
K GetPaneHeight

$#+K! cbDockPane::GetRealRect

wxRect& GetRealRect()K

Returns the bounds of the pane, in parent coordinates.

---

cbDockPane::GetRealRect
cbdockpanegetrealrect
browse00126
K cbDockPane GetRealRect
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbdockpane')")
K GetRealRect

$#+K!**cbDockPane::GetRow**

cbRowInfo* GetRow(**int** *row*)K

protected really (accessed only by plugins) Returns the row info for a row index. Internal function called by plugins.

---

cbDockPane::GetRow
cbdockpanegetrow
browse00127
K cbDockPane GetRow
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbdockpane')")
K GetRow

$#+K!cbDockPane::GetRowAt

int GetRowAt(**int** *paneY*)K

Returns the row at the given vertical position. Returns -1 if the row is not present at given vertical position. Internal function called by plugins.

int GetRowAt(**int** *upperY*, **int** *lowerY*)K

Returns the row between the given vertical positions. Returns -1 if the row is not present. Internal function called by plugins.

---

cbDockPane::GetRowAt
cbdockpanegetrowat
browse00128
K cbDockPane GetRowAt
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbdockpane')")
K GetRowAt
K GetRowAt

$#+K!cbDockPane::GetRowIndex

int GetRowIndex(**cbRowInfo*** *pRow*)K

Returns the row index for the given row info. Internal function called by plugins.

---

cbDockPane::GetRowIndex
cbdockpanegetrowindex
browse00129
K cbDockPane GetRowIndex
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbdockpane')")
K GetRowIndex

$#+K!cbDockPane::GetRowList

RowArrayT& GetRowList()K

Returns an array of rows. Used by updates-managers.

---

cbDockPane::GetRowList
cbdockpanegetrowlist
browse00130
K cbDockPane GetRowList
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbdockpane')")
K GetRowList

$#+K!**cbDockPane::GetRowResizeRange**

void GetRowResizeRange(**cbRowInfo*** *pRow*, **int*** *from*, **int*** *till*, **bool** *forUpperHandle*)K

Returns the row's resize range.

---

cbDockPane::GetRowResizeRange
cbdockpanegetrowresizerange
browse00131
K cbDockPane GetRowResizeRange
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbdockpane')")
K GetRowResizeRange

$#+K!cbDockPane::GetRowShapeData

void GetRowShapeData(**cbRowInfo*** *pRow*, **wxList*** *pLst*)[K]

Returns row shape data. cbBarShapeData objects will be added to the given pLst. cbBarShapeData is used for storing the original bar's positions in the row, when the 'non-destructive-friction' option is turned on.

---

[c]bDockPane::GetRowShapeData
[c]bdockpanegetrowshapedata
[b]rowse00132
[K] cbDockPane GetRowShapeData
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbdockpane')")
[K] GetRowShapeData

$#+K!**cbDockPane::GetRowY**

int GetRowY(**cbRowInfo*** *pRow*)K

Gets the vertical position at the given row. Internal function called by plugins.

---

cbDockPane::GetRowY
cbdockpanegetrowy
browse00133
K cbDockPane GetRowY
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbdockpane')")
K GetRowY

$#+K!cbDockPane::HasNotFixedBarsLeft

bool HasNotFixedBarsLeft(**cbBarInfo*** *pBar*)K

Returns TRUE if there are any variable-sized rows to the left of this one. Internal function called by plugins.

---

cbDockPane::HasNotFixedBarsLeft
cbdockpanehasnotfixedbarsleft
browse00134
K cbDockPane HasNotFixedBarsLeft
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbdockpane')")
K HasNotFixedBarsLeft

$#+K!cbDockPane::HasNotFixedBarsRight

bool HasNotFixedBarsRight(**cbBarInfo*** *pBar*)[K]

Returns TRUE if there are any variable-sized rows to the right of this one. Internal function called by plugins.

---

[c]bDockPane::HasNotFixedBarsRight
[c]bdockpanehasnotfixedbarsright
[b]rowse00135
[K] cbDockPane HasNotFixedBarsRight
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbdockpane')")
[K] HasNotFixedBarsRight

$#+K!cbDockPane::HasNotFixedRowsAbove

bool HasNotFixedRowsAbove(**cbRowInfo*** *pRow*)[K]

Returns TRUE if there are any variable-sized rows above this one. Internal function called by plugins.

---

[c]bDockPane::HasNotFixedRowsAbove
[c]bdockpanehasnotfixedrowsabove
[b]rowse00136
[K] cbDockPane HasNotFixedRowsAbove
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbdockpane')")
[K] HasNotFixedRowsAbove

$#+K!cbDockPane::HasNotFixedRowsBelow

bool HasNotFixedRowsBelow(**cbRowInfo*** *pRow*)K

Returns TRUE if there are any variable-sized rows below this one. Internal function called by plugins.

---

cbDockPane::HasNotFixedRowsBelow
cbdockpanehasnotfixedrowsbelow
browse00137
K cbDockPane HasNotFixedRowsBelow
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbdockpane')")
K HasNotFixedRowsBelow

$#+K!cbDockPane::HasPoint

bool HasPoint(**const wxPoint&** *pos*, **int** *x*, **int** *y*, **int** *width*, **int** *height*)[K]

Returns TRUE if pos is within the given rectangle. Internal function called by plugins.

---

[c]bDockPane::HasPoint
[c]bdockpanehaspoint
[b]rowse00138
[K] cbDockPane HasPoint
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbdockpane')")
[K] HasPoint

$#+K!cbDockPane::HitTestPaneItems

int HitTestPaneItems(**const wxPoint&** *pos*, **cbRowInfo**** *ppRow*, **cbBarInfo**** *ppBar*)[K]

Returns the result of hit-testing items in the pane. See CB_HITTEST_RESULT enumerated type. pos is the position in this pane's coordinates.

---

[c]bDockPane::HitTestPaneItems
[c]bdockpanehittestpaneitems
[b]rowse00139
[K] cbDockPane HitTestPaneItems
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbdockpane')")
[K] HitTestPaneItems

$#+K!cbDockPane::InitLinksForRow

void InitLinksForRow(**cbRowInfo*** *pRow*)K

Sets up links between bars. Internal function called by plugins.

---

cbDockPane::InitLinksForRow
cbdockpaneinitlinksforrow
browse00140
K cbDockPane InitLinksForRow
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbdockpane')")
K InitLinksForRow

$#+K!cbDockPane::InitLinksForRows

void InitLinksForRows()K

Sets up links between bars. Internal function called by plugins.

---

cbDockPane::InitLinksForRows
cbdockpaneinitlinksforrows
browse00141
K cbDockPane InitLinksForRows
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbdockpane')")
K InitLinksForRows

$#+K!cbDockPane::InsertBar

void InsertBar(**cbBarInfo*** *pBarInfo*)K

Inserts bar and sets its position according to the preferred settings given in pBarInfo.

void InsertBar(**cbBarInfo*** *pBar*, **const wxRect&** *rect*)K

Inserts the bar into this pane. rect is given in the parent frame's coordinates.

void InsertBar(**cbBarInfo*** *pBar*, **cbRowInfo*** *pIntoRow*)K

Inserts the bar into the given row, with dimensions and position stored in pBarInfo->mBounds. Returns the node of inserted bar.

---

cbDockPane::InsertBar
cbdockpaneinsertbar
browse00142
K cbDockPane InsertBar
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbdockpane')")
K InsertBar
K InsertBar
K InsertBar

$#+K!cbDockPane::InsertRow

void InsertRow(**cbRowInfo*** *pRow*, **cbRowInfo*** *pBeforeRow*)K

Inserts a row. Does not refresh the inserted row immediately. If pBeforeRowNode is NULL, the row is appended to the end of pane's row list.

---

cbDockPane::InsertRow
cbdockpaneinsertrow
browse00143
K cbDockPane InsertRow
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbdockpane')")
K InsertRow

$#+K!cbDockPane::IsFixedSize

bool IsFixedSize(**cbBarInfo*** *pInfo*)K

Returns TRUE if the bar's dimension information indicates a fixed size. Internal function called by plugins.

---

cbDockPane::IsFixedSize
cbdockpaneisfixedsize
browse00144
K cbDockPane IsFixedSize
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbdockpane')")
K IsFixedSize

$#+K!cbDockPane::IsHorizontal

bool IsHorizontal()K

Returns TRUE if the pane is aligned to the top or bottom.

---

cbDockPane::IsHorizontal
cbdockpaneishorizontal
browse00145
K cbDockPane IsHorizontal
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbdockpane')")
K IsHorizontal

$#+K! cbDockPane::MatchesMask

bool MatchesMask(**int** *paneMask*)K

Returns TRUE if the given mask matches the pane's mask.

---

c bDockPane::MatchesMask
c bdockpanematchesmask
b rowse00146
K cbDockPane MatchesMask
E nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbdockpane')")
K MatchesMask

$#+K!**cbDockPane::PaintBar**

void PaintBar(**cbBarInfo*** *pBar*, **wxDC&** *dc*)K

Calls PaintBarDecorations and PaintBarHandles. Internal function called by plugins.

---

cbDockPane::PaintBar
cbdockpanepaintbar
browse00147
K cbDockPane PaintBar
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbdockpane')")
K PaintBar

$#+K!cbDockPane::PaintBarDecorations

void PaintBarDecorations(**cbBarInfo*** *pBar*, **wxDC&** *dc*)[K]

protected really (accessed only by plugins) Generates a cbDrawBarDecorEvent and sends it to the layout to paint the bar decorations. Internal function called by plugins.

---

[c]bDockPane::PaintBarDecorations
[c]bdockpanepaintbardecorations
[b]rowse00148
[K] cbDockPane PaintBarDecorations
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbdockpane')")
[K] PaintBarDecorations

$#+K!cbDockPane::PaintBarHandles

void PaintBarHandles(**cbBarInfo*** *pBar*, **wxDC&** *dc*)[K]

Generates a cbDrawBarHandlesEvent and sends it to the layout to paint the bar handles. Internal function called by plugins.

---

[c]bDockPane::PaintBarHandles
[c]bdockpanepaintbarhandles
[b]rowse00149
[K] cbDockPane PaintBarHandles
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbdockpane')")
[K] PaintBarHandles

$#+K!**cbDockPane::PaintPane**

void PaintPane(**wxDC&** *dc*)K

Paints the pane background, the row background and decorations, and finally the pane decorations. Internal function called by plugins.

---

cbDockPane::PaintPane
cbdockpanepaintpane
browse00150
K cbDockPane PaintPane
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbdockpane')")
K PaintPane

$#+K!cbDockPane::PaintPaneBackground

void PaintPaneBackground(**wxDC&** *dc*)[K]

Generates cbDrawPaneBkGroundEvent and sends it to the layout. Internal function called by plugins.

---

[c]bDockPane::PaintPaneBackground
[c]bdockpanepaintpanebackground
[b]rowse00151
[K] cbDockPane PaintPaneBackground
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbdockpane')")
[K] PaintPaneBackground

$#+K!cbDockPane::PaintPaneDecorations

void PaintPaneDecorations(**wxDC&** *dc*)K

Generates cbDrawPaneDecorEvent and sends it to the layout. Internal function called by plugins.

---

cbDockPane::PaintPaneDecorations
cbdockpanepaintpanedecorations
browse00152
K cbDockPane PaintPaneDecorations
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbdockpane')")
K PaintPaneDecorations

$#+K!**cbDockPane::PaintRow**

void PaintRow(**cbRowInfo*** *pRow*, **wxDC&** *dc*)K

Calls PaintRowBackground, PaintRowDecorations, PaintRowHandles. Internal function called by plugins.

---

cbDockPane::PaintRow
cbdockpanepaintrow
browse00153
K cbDockPane PaintRow
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbdockpane')")
K PaintRow

$#+K!cbDockPane::PaintRowBackground

void PaintRowBackground(**cbRowInfo*** *pRow*, **wxDC&** *dc*)[K]

Generates cbDrawRowBkGroundEvent and sends it to the layout. Internal function called by plugins.

---

[c]bDockPane::PaintRowBackground
[c]bdockpanepaintrowbackground
[b]rowse00154
[K] cbDockPane PaintRowBackground
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbdockpane')")
[K] PaintRowBackground

$#+K!cbDockPane::PaintRowDecorations

void PaintRowDecorations(**cbRowInfo*** *pRow*, **wxDC&** *dc*)K

Calls PaintBarDecorations for each row. Internal function called by plugins.

---

cbDockPane::PaintRowDecorations
cbdockpanepaintrowdecorations
browse00155
K cbDockPane PaintRowDecorations
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbdockpane')")
K PaintRowDecorations

$#+K!cbDockPane::PaintRowHandles

void PaintRowHandles(**cbRowInfo*** *pRow*, **wxDC&** *dc*)[K]

Generates cbDrawRowHandlesEvent and cbDrawRowDecorEvent and sends them to the layout. Internal function called by plugins.

---

[c]bDockPane::PaintRowHandles
[c]bdockpanepaintrowhandles
[b]rowse00156
[K] cbDockPane PaintRowHandles
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbdockpane')")
[K] PaintRowHandles

$#+K!**cbDockPane::PaneToFrame**

void PaneToFrame(**wxRect*** *pRect*)K

Coordinate translation between parent's frame and this pane. Internal function called by plugins.

void PaneToFrame(**int*** *x*, **int*** *y*)K

Coordinate translation between parent's frame and this pane. Internal function called by plugins.

---

cbDockPane::PaneToFrame
cbdockpanepanetoframe
browse00157
K cbDockPane PaneToFrame
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbdockpane')")
K PaneToFrame
K PaneToFrame

$#+K!**cbDockPane::RecalcLayout**

void RecalcLayout()K

Generates events to perform layout calculations.

---

cbDockPane::RecalcLayout
cbdockpanerecalclayout
browse00158
K cbDockPane RecalcLayout
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbdockpane')")
K RecalcLayout

$#+K!cbDockPane::RecalcRowLayout

void RecalcRowLayout(**cbRowInfo*** *pRow*)[K]

Generates a cbLayoutRowEvent event to recalculate row layouts. Internal function called by plugins.

---

[c]bDockPane::RecalcRowLayout
[c]bdockpanerecalcrowlayout
[b]rowse00159
[K] cbDockPane RecalcRowLayout
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbdockpane')")
[K] RecalcRowLayout

$#+K!cbDockPane::RemoveBar

void RemoveBar(**cbBarInfo*** *pBar*)K

Removes the bar from this pane. Does not destroy the bar.

---

cbDockPane::RemoveBar
cbdockpaneremovebar
browse00160
K cbDockPane RemoveBar
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbdockpane')")
K RemoveBar

$#+K!**cbDockPane::RemoveRow**

void RemoveRow(**cbRowInfo*** *pRow*)K

Removes the row from this pane. Does not destroy the row object.

---

cbDockPane::RemoveRow
cbdockpaneremoverow
browse00161
K cbDockPane RemoveRow
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbdockpane')")
K RemoveRow

$#+K!**cbDockPane::ResizeBar**

void ResizeBar(**cbBarInfo*** *pBar*, **int** *ofs*, **bool** *forLeftHandle*)K

Row/bar resizing related helper-method.

---

cbDockPane::ResizeBar
cbdockpaneresizebar
browse00162
K cbDockPane ResizeBar
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbdockpane')")
K ResizeBar

$#+K!**cbDockPane::ResizeRow**

void ResizeRow(**cbRowInfo*** *pRow*, **int** *ofs*, **bool** *forUpperHandle*)K

Row/bar resizing related helper-method.

---

cbDockPane::ResizeRow
cbdockpaneresizerow
browse00163
K cbDockPane ResizeRow
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbdockpane')")
K ResizeRow

$#+K!cbDockPane::SetBoundsInParent

void SetBoundsInParent(**const wxRect&** *rect*)K

Set the position and dimensions of the pane in the parent frame's coordinates.

---

cbDockPane::SetBoundsInParent
cbdockpanesetboundsinparent
browse00164
K cbDockPane SetBoundsInParent
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbdockpane')")
K SetBoundsInParent

$#+K! cbDockPane::SetMargins

void SetMargins(**int** *top*, **int** *bottom*, **int** *left*, **int** *right*)K

Sets pane's margins in frame's coordinate orientations.

---

cbDockPane::SetMargins
cbdockpanesetmargins
browse00165
K cbDockPane SetMargins
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbdockpane')")
K SetMargins

$#+K!cbDockPane::SetPaneWidth

void SetPaneWidth(**int** *width*)K

Sets pane's width in the pane's coordinates (including margins).

---

cbDockPane::SetPaneWidth
cbdockpanesetpanewidth
browse00166
K cbDockPane SetPaneWidth
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbdockpane')")
K SetPaneWidth

$#+K!cbDockPane::SetRowHeight

void SetRowHeight(**cbRowInfo*** *pRow*, **int** *newHeight*)K

Sets the row height for the given height. newHeight includes the height of row handles, if present. Internal function called by plugins.

---

cbDockPane::SetRowHeight
cbdockpanesetrowheight
browse00167
K cbDockPane SetRowHeight
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbdockpane')")
K SetRowHeight

$#+K!cbDockPane::SetRowShapeData

void SetRowShapeData(**cbRowInfo*** *pRowNode*, **wxList*** *pLst*)[K]

Sets the shape data for the given row, using the data provided in pLst. cbBarShapeData is used for storing the original bar's positions in the row, when the 'non-destructive-friction' option is turned on.

---

[c]bDockPane::SetRowShapeData
[c]bdockpanesetrowshapedata
[b]rowse00168
[K] cbDockPane SetRowShapeData
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbdockpane')")
[K] SetRowShapeData

$#+K!cbDockPane::SizeBar

void SizeBar(**cbBarInfo*** *pBar*)K

Generates a cbSizeBarWndEvent and sends it to the layout. Internal function called by plugins.

---

cbDockPane::SizeBar
cbdockpanesizebar
browse00169
K cbDockPane SizeBar
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbdockpane')")
K SizeBar

$#+K!cbDockPane::SizePaneObjects

void SizePaneObjects()K

Calls SizeRowObjects for each row. Internal function called by plugins.

---

cbDockPane::SizePaneObjects
cbdockpanesizepaneobjects
browse00170
K cbDockPane SizePaneObjects
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbdockpane')")
K SizePaneObjects

$#+K!cbDockPane::SizeRowObjects

void SizeRowObjects(**cbRowInfo*** *pRow*)K

Calls SizeBar for each bar in the row. Internal function called by plugins.

---

cbDockPane::SizeRowObjects
cbdockpanesizerowobjects
browse00171
K cbDockPane SizeRowObjects
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbdockpane')")
K SizeRowObjects

$#+K!cbDockPane::StartDrawInArea

wxDC* StartDrawInArea(**const wxRect&** *area*)K

Generates cbStartDrawInAreaEvent and sends it to the layout. Internal function called by plugins.

---

cbDockPane::StartDrawInArea
cbdockpanestartdrawinarea
browse00172
K cbDockPane StartDrawInArea
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbdockpane')")
K StartDrawInArea

$#+K!cbDockPane::SyncRowFlags

void SyncRowFlags(**cbRowInfo*** *pRow*)[K]

Sets up flags in the row information structure, so that they match the changed state of row items correctly. Internal function called by plugins.

---

[c]bDockPane::SyncRowFlags
[c]bdockpanesyncrowflags
[b]rowse00173
[K] cbDockPane SyncRowFlags
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbdockpane')")
[K] SyncRowFlags

$#+K!cbDrawBarDecorEvent::cbDrawBarDecorEvent

cbDrawBarDecorEvent(**cbBarInfo*** *pBar*, **wxDC&** *dc*, **cbDockPane*** *pPane*)K

Constructor, taking bar information, device context, and pane.

---

cbDrawBarDecorEvent::cbDrawBarDecorEvent
cbdrawbardecoreventcbdrawbardecorevent
browse00175
K cbDrawBarDecorEvent cbDrawBarDecorEvent
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbdrawbardecorevent')")
K cbDrawBarDecorEvent

$#+K!**cbDrawBarHandlesEvent::cbDrawBarHandlesEvent**

cbDrawBarHandlesEvent(**cbBarInfo*** *pBar*, **wxDC&** *dc*, **cbDockPane*** *pPane*)K

Constructor, taking bar information, device context, and pane.

---

cbDrawBarHandlesEvent::cbDrawBarHandlesEvent
cbdrawbarhandleseventcbdrawbarhandlesevent
browse00177
K cbDrawBarHandlesEvent cbDrawBarHandlesEvent
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbdrawbarhandlesevent')")
K cbDrawBarHandlesEvent

$#+K!cbDrawHintRectEvent::cbDrawHintRectEvent

cbDrawHintRectEvent(**const wxRect&** *rect*, **bool** *isInClient*, **bool** *eraseRect*, **bool** *lastTime*)[K]

e.g. with fat/hatched border Constructor, taking hint rectangle and three flags.

---

[c]bDrawHintRectEvent::cbDrawHintRectEvent
[c]bdrawhintrecteventcbdrawhintrectevent
[b]rowse00179
[K] cbDrawHintRectEvent cbDrawHintRectEvent
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbdrawhintrectevent')")
[K] cbDrawHintRectEvent

$#+K!cbDrawPaneBkGroundEvent::cbDrawPaneBkGroundEvent

cbDrawPaneBkGroundEvent(**wxDC&** *dc*, **cbDockPane*** *pPane*)[K]

Constructor, taking device context and pane.

---

[c]bDrawPaneBkGroundEvent::cbDrawPaneBkGroundEvent
[c]bdrawpanebkgroundeventcbdrawpanebkgroundevent
[b]rowse00181
[K] cbDrawPaneBkGroundEvent cbDrawPaneBkGroundEvent
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbdrawpanebkgroundevent')")
[K] cbDrawPaneBkGroundEvent

$#+K!**cbDrawPaneDecorEvent::cbDrawPaneDecorEvent**

cbDrawPaneDecorEvent(**wxDC&** *dc*, **cbDockPane*** *pPane*)[K]

Constructor, taking device context and pane.

---

[c]bDrawPaneDecorEvent::cbDrawPaneDecorEvent
[c]bdrawpanedecoreventcbdrawpanedecorevent
[b]rowse00183
[K] cbDrawPaneDecorEvent cbDrawPaneDecorEvent
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbdrawpanedecorevent')")
[K] cbDrawPaneDecorEvent

$#+K!**cbDrawRowBkGroundEvent::cbDrawRowBkGroundEvent**

cbDrawRowBkGroundEvent(**cbRowInfo*** *pRow*, **wxDC&** *dc*, **cbDockPane*** *pPane*)[K]

Constructor, taking row information, device context, and pane.

---

[c]bDrawRowBkGroundEvent::cbDrawRowBkGroundEvent
[c]bdrawrowbkgroundeventcbdrawrowbkgroundevent
[b]rowse00185
[K] cbDrawRowBkGroundEvent cbDrawRowBkGroundEvent
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbdrawrowbkgroundevent')")
[K] cbDrawRowBkGroundEvent

$#+K!cbDrawRowDecorEvent::cbDrawRowDecorEvent

cbDrawRowDecorEvent(**cbRowInfo*** *pRow*, **wxDC&** *dc*, **cbDockPane*** *pPane*)[K]

Constructor, taking row information, device context, and pane.

---

[c]bDrawRowDecorEvent::cbDrawRowDecorEvent
[c]bdrawrowdecoreventcbdrawrowdecorevent
[b]rowse00187
[K] cbDrawRowDecorEvent cbDrawRowDecorEvent
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbdrawrowdecorevent')")
[K] cbDrawRowDecorEvent

$#+K!**cbDrawRowHandlesEvent::cbDrawRowHandlesEvent**

cbDrawRowHandlesEvent(**cbRowInfo*** *pRow*, **wxDC&** *dc*, **cbDockPane*** *pPane*)K

Constructor, taking row information, device context, and pane.

---

cbDrawRowHandlesEvent::cbDrawRowHandlesEvent
cbdrawrowhandleseventcbdrawrowhandlesevent
browse00189
K cbDrawRowHandlesEvent cbDrawRowHandlesEvent
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbdrawrowhandlesevent')")
K cbDrawRowHandlesEvent

$#+K!cbDynToolBarDimHandler::OnChangeBarState

void OnChangeBarState(**cbBarInfo*** *pBar*, **int** *newState*)[K]

Called when the bar changes state.

---

[c]bDynToolBarDimHandler::OnChangeBarState
[c]bdyntoolbardimhandleronchangebarstate
[b]rowse00191
[K] cbDynToolBarDimHandler OnChangeBarState
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbdyntoolbardimhandler')")
[K] OnChangeBarState

$#+K!cbDynToolBarDimHandler::OnResizeBar

void OnResizeBar(**cbBarInfo*** *pBar*, **const wxSize&** *given*, **wxSize&** *preferred*)K

Called when a bar is resized.

---

cbDynToolBarDimHandler::OnResizeBar
cbdyntoolbardimhandleronresizebar
browse00192
K cbDynToolBarDimHandler OnResizeBar
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbdyntoolbardimhandler')")
K OnResizeBar

$#+K!**cbFinishDrawInAreaEvent::cbFinishDrawInAreaEvent**

cbFinishDrawInAreaEvent(**const wxRect&** *area*, **cbDockPane*** *pPane*)[K]

Constructor, taking rectangular area and pane.

---

[c]bFinishDrawInAreaEvent::cbFinishDrawInAreaEvent
[c]bfinishdrawinareaeventcbfinishdrawinareaevent
[b]rowse00194
[K] cbFinishDrawInAreaEvent cbFinishDrawInAreaEvent
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbfinishdrawinareaevent')")
[K] cbFinishDrawInAreaEvent

$#+K!**cbFloatedBarWindow::cbFloatedBarWindow**

cbFloatedBarWindow()[K]

Default constructor.

---

[c]bFloatedBarWindow::cbFloatedBarWindow
[c]bfloatedbarwindowcbfloatedbarwindow
[b]rowse00196
[K] cbFloatedBarWindow cbFloatedBarWindow
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbfloatedbarwindow')")
[K] cbFloatedBarWindow

$#+K!cbFloatedBarWindow::GetBar

cbBarInfo* GetBar()K

Returns the bar information for this window.

---

cbFloatedBarWindow::GetBar
cbfloatedbarwindowgetbar
browse00197
K cbFloatedBarWindow GetBar
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbfloatedbarwindow')")
K GetBar

$#+K!cbFloatedBarWindow::GetPreferredSize

wxSize GetPreferredSize(**const wxSize&** *given*)K

Overridden function returning the preferred size.

---

cbFloatedBarWindow::GetPreferredSize
cbfloatedbarwindowgetpreferredsize
browse00198
K cbFloatedBarWindow GetPreferredSize
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbfloatedbarwindow')")
K GetPreferredSize

$#+K!cbFloatedBarWindow::HandleTitleClick

bool HandleTitleClick(**wxMouseEvent&** *event*)K

Overridden function responding to mouse button clicks on the titlebar.

---

cbFloatedBarWindow::HandleTitleClick
cbfloatedbarwindowhandletitleclick
browse00199
K cbFloatedBarWindow HandleTitleClick
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbfloatedbarwindow')")
K HandleTitleClick

$#+K!**cbFloatedBarWindow::OnDblClick**

void OnDblClick(**wxMouseEvent&** *event*)K

Responds to double-click mouse events.

---

cbFloatedBarWindow::OnDblClick
cbfloatedbarwindowondblclick
browse00200
K cbFloatedBarWindow OnDblClick
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbfloatedbarwindow')")
K OnDblClick

$#+K!cbFloatedBarWindow::OnMiniButtonClicked

void OnMiniButtonClicked(**int** *btnIdx*)K

Overridden function responding to mouse clicks on mini-buttons.

---

cbFloatedBarWindow::OnMiniButtonClicked
cbfloatedbarwindowonminibuttonclicked
browse00201
K cbFloatedBarWindow OnMiniButtonClicked
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbfloatedbarwindow')")
K OnMiniButtonClicked

$#+K!**cbFloatedBarWindow::PositionFloatedWnd**

void PositionFloatedWnd(**int** *scrX*, **int** *scrY*, **int** *width*, **int** *height*)K

Position the floating window. The given coordinates are those of the bar itself; the floated container window's position and size are ajusted accordingly.

---

cbFloatedBarWindow::PositionFloatedWnd
cbfloatedbarwindowpositionfloatedwnd
browse00202
K cbFloatedBarWindow PositionFloatedWnd
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbfloatedbarwindow')")
K PositionFloatedWnd

$#+K!cbFloatedBarWindow::SetBar

void SetBar(**cbBarInfo*** *pBar*)K

Sets the bar information for this window.

---

cbFloatedBarWindow::SetBar
cbfloatedbarwindowsetbar
browse00203
K cbFloatedBarWindow SetBar
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbfloatedbarwindow')")
K SetBar

$#+K!**cbFloatedBarWindow::SetLayout**

void SetLayout(**wxFrameLayout*** *pLayout*)K

Sets the layout for this window.

---

cbFloatedBarWindow::SetLayout
cbfloatedbarwindowsetlayout
browse00204
K cbFloatedBarWindow SetLayout
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbfloatedbarwindow')")
K SetLayout

$#+K!cbGCUpdatesMgr::cbGCUpdatesMgr

cbGCUpdatesMgr()[K]

Default constructor.

cbGCUpdatesMgr(**wxFrameLayout*** *pPanel*)[K]

Constructor, taking a frame layout.

---

[c]bGCUpdatesMgr::cbGCUpdatesMgr
[c]bgcupdatesmgrcbgcupdatesmgr
[b]rowse00206
[K] cbGCUpdatesMgr cbGCUpdatesMgr
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbgcupdatesmgr')")
[K] cbGCUpdatesMgr
[K] cbGCUpdatesMgr

$#+K!**cbGCUpdatesMgr::AddItem**

void AddItem(**wxList&** *itemList*, **cbBarInfo*** *pBar*, **cbDockPane*** *pPane*, **wxRect&** *curBounds*, **wxRect&** *prevBounds*)K

Internal function for repositioning items.

---

cbGCUpdatesMgr::AddItem
cbgcupdatesmgradditem
browse00207
K cbGCUpdatesMgr AddItem
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbgcupdatesmgr')")
K AddItem

$#+K!cbGCUpdatesMgr::DoRepositionItems

void DoRepositionItems(**wxList&** *items*)[K]

Internal function for repositioning items.

---

[c]bGCUpdatesMgr::DoRepositionItems
[c]bgcupdatesmgrdorepositionitems
[b]rowse00208
[K] cbGCUpdatesMgr DoRepositionItems
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbgcupdatesmgr')")
[K] DoRepositionItems

$#+K!cbGCUpdatesMgr::OnStartChanges

void OnStartChanges()K

Receives notifications from the frame layout.

---

cbGCUpdatesMgr::OnStartChanges
cbgcupdatesmgronstartchanges
browse00209
K cbGCUpdatesMgr OnStartChanges
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbgcupdatesmgr')")
K OnStartChanges

$#+K!cbGCUpdatesMgr::UpdateNow

void UpdateNow()[K]

Refreshes the parts of the frame layout which need an update.

---

[c]bGCUpdatesMgr::UpdateNow
[c]bgcupdatesmgrupdatenow
[b]rowse00210
[K] cbGCUpdatesMgr UpdateNow
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbgcupdatesmgr')")
[K] UpdateNow

$#+K!cbHintAnimationPlugin::cbHintAnimationPlugin

cbHintAnimationPlugin()[K]

Default constructor.

cbHintAnimationPlugin(**wxFrameLayout*** *pPanel*, **int** *paneMask* = *wxALL_PANES*)[K]

Constructor, taking a layout panel and pane mask.

---

[c]bHintAnimationPlugin::cbHintAnimationPlugin
[c]bhintanimationplugincbhintanimationplugin
[b]rowse00212
[K] cbHintAnimationPlugin cbHintAnimationPlugin
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbhintanimationplugin')")
[K] cbHintAnimationPlugin
[K] cbHintAnimationPlugin

$#+K!cbHintAnimationPlugin::~cbHintAnimationPlugin

~cbHintAnimationPlugin()K

Destructor.

---

cbHintAnimationPlugin::~cbHintAnimationPlugin
cbhintanimationplugindtor
browse00213
K cbHintAnimationPlugin ~cbHintAnimationPlugin
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbhintanimationplugin')")
K ~cbHintAnimationPlugin

$#+K!**cbHintAnimationPlugin::DoDrawHintRect**

void DoDrawHintRect(**wxRect&** *rect*, **bool** *isInClientRect*)[K]

Internal function for drawing a hint rectangle.

---

[c]bHintAnimationPlugin::DoDrawHintRect
[c]bhintanimationplugindodrawhintrect
[b]rowse00214
[K] cbHintAnimationPlugin DoDrawHintRect
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbhintanimationplugin')")
[K] DoDrawHintRect

$#+K!**cbHintAnimationPlugin::DrawHintRect**

void DrawHintRect(**wxRect&** *rect*, **bool** *isInClientRect*)K

Internal function for drawing a hint rectangle.

---

cbHintAnimationPlugin::DrawHintRect
cbhintanimationplugindrawhintrect
browse00215
K cbHintAnimationPlugin DrawHintRect
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbhintanimationplugin')")
K DrawHintRect

$#+K!cbHintAnimationPlugin::EraseHintRect

void EraseHintRect(**wxRect&** *rect*, **bool** *isInClientRect*)K

Internal function for erasing a hint rectangle.

---

cbHintAnimationPlugin::EraseHintRect
cbhintanimationpluginerasehintrect
browse00216
K cbHintAnimationPlugin EraseHintRect
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbhintanimationplugin')")
K EraseHintRect

$#+K!**cbHintAnimationPlugin::FinishTracking**

void FinishTracking()K

Internal function for finishing tracking.

---

cbHintAnimationPlugin::FinishTracking
cbhintanimationpluginfinishtracking
browse00217
K cbHintAnimationPlugin FinishTracking
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbhintanimationplugin')")
K FinishTracking

$#+K!cbHintAnimationPlugin::OnDrawHintRect

void OnDrawHintRect(**cbDrawHintRectEvent&** *event*)K

Event handler respoding to hint draw events.

---

cbHintAnimationPlugin::OnDrawHintRect
cbhintanimationpluginondrawhintrect
browse00218
K cbHintAnimationPlugin OnDrawHintRect
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbhintanimationplugin')")
K OnDrawHintRect

$#+K!**cbHintAnimationPlugin::RectToScr**

void RectToScr(**wxRect&** *frameRect*, **wxRect&** *scrRect*)K

Internal function for translating coordinates.

---

cbHintAnimationPlugin::RectToScr
cbhintanimationpluginrecttoscr
browse00219
K cbHintAnimationPlugin RectToScr
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbhintanimationplugin')")
K RectToScr

$#+K!cbHintAnimationPlugin::StartTracking

void StartTracking()[K]

speed is constant. Default: TRUE TBD:: get/set methods for above members Internal function for starting tracking.

---

[c]bHintAnimationPlugin::StartTracking
[c]bhintanimationpluginstarttracking
[b]rowse00220
[K] cbHintAnimationPlugin StartTracking
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbhintanimationplugin')")
[K] StartTracking

$#+K!cbInsertBarEvent::cbInsertBarEvent

cbInsertBarEvent(**cbBarInfo*** *pBar*, **cbRowInfo*** *pIntoRow*, **cbDockPane*** *pPane*)K

Constructor, taking bar information, row information, and pane.

---

cbInsertBarEvent::cbInsertBarEvent
cbinsertbareventcbinsertbarevent
browse00222
K cbInsertBarEvent cbInsertBarEvent
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbinsertbarevent')")
K cbInsertBarEvent

$#+K!cbLayoutRowEvent::cbLayoutRowEvent

cbLayoutRowEvent(**cbRowInfo*** *pRow*, **cbDockPane*** *pPane*)K

Constructor, taking row information and pane.

---

cbLayoutRowEvent::cbLayoutRowEvent
cblayoutroweventcblayoutrowevent
browse00224
K cbLayoutRowEvent cbLayoutRowEvent
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cblayoutrowevent')")
K cbLayoutRowEvent

$#+K!**cbLayoutRowsEvent::cbLayoutRowsEvent**

cbLayoutRowsEvent(**cbDockPane*** *pPane*)K

Constructor, taking pane.

---

cbLayoutRowsEvent::cbLayoutRowsEvent
cblayoutrowseventcblayoutrowsevent
browse00226
K cbLayoutRowsEvent cbLayoutRowsEvent
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cblayoutrowsevent')")
K cbLayoutRowsEvent

$#+K!cbLeftDClickEvent::cbLeftDClickEvent

cbLeftDClickEvent(**const wxPoint&** *pos*, **cbDockPane*** *pPane*)[K]

Constructor, taking mouse position and pane.

---

[c]bLeftDClickEvent::cbLeftDClickEvent
[c]bleftdclickeventcbleftdclickevent
[b]rowse00228
[K] cbLeftDClickEvent cbLeftDClickEvent
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbleftdclickevent')")
[K] cbLeftDClickEvent

$#+K!**cbLeftDownEvent::cbLeftDownEvent**

cbLeftDownEvent(**const wxPoint&** *pos*, **cbDockPane*** *pPane*)K

Constructor, taking mouse position and pane.

---

cbLeftDownEvent::cbLeftDownEvent
cbleftdowneventcbleftdownevent
browse00230
K cbLeftDownEvent cbLeftDownEvent
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbleftdownevent')")
K cbLeftDownEvent

$#+K!cbLeftUpEvent::cbLeftUpEvent

cbLeftUpEvent(**const wxPoint&** *pos*, **cbDockPane*** *pPane*)[K]

Constructor, taking mouse position and pane.

---

[c]bLeftUpEvent::cbLeftUpEvent
[c]bleftupeventcbleftupevent
[b]rowse00232
[K] cbLeftUpEvent cbLeftUpEvent
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbleftupevent')")
[K] cbLeftUpEvent

$#+K!**cbMiniButton::cbMiniButton**

cbMiniButton()[K]

Default constructor.

---

[c]bMiniButton::cbMiniButton
[c]bminibuttoncbminibutton
[b]rowse00234
[K] cbMiniButton cbMiniButton
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbminibutton')")
[K] cbMiniButton

$#+K!**cbMiniButton::Draw**

void Draw(**wxDC&** *dc*)[K]

Draws the button. Override this to implement the desired appearance.

---

[c]bMiniButton::Draw
[c]bminibuttondraw
[b]rowse00235
[K] cbMiniButton Draw
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbminibutton')")
[K] Draw

$#+K!**cbMiniButton::Enable**

void Enable(**bool** *enable*)K

Enable or disable the button.

---

cbMiniButton::Enable
cbminibuttonenable
browse00236
K cbMiniButton Enable
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbminibutton')")
K Enable

$#+K!cbMiniButton::HitTest

bool HitTest(**const wxPoint&** *pos*)K

Returns TRUE if the given position was over the button.

---

cbMiniButton::HitTest
cbminibuttonhittest
browse00237
K cbMiniButton HitTest
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbminibutton')")
K HitTest

$#+K!**cbMiniButton::IsPressed**

bool IsPressed()K

Returns TRUE if this button is pressed.

---

cbMiniButton::IsPressed
cbminibuttonispressed
browse00238
K cbMiniButton IsPressed
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbminibutton')")
K IsPressed

$#+K!cbMiniButton::OnLeftDown

void OnLeftDown(**const wxPoint&** *pos*)K

Responds to a left down event.

---

cbMiniButton::OnLeftDown
cbminibuttononleftdown
browse00239
K cbMiniButton OnLeftDown
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbminibutton')")
K OnLeftDown

$#+K!**cbMiniButton::OnLeftUp**

void OnLeftUp(**const wxPoint&** *pos*)K

Responds to a left up event.

---

cbMiniButton::OnLeftUp
cbminibuttononleftup
browse00240
K cbMiniButton OnLeftUp
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbminibutton')")
K OnLeftUp

$#+K!**cbMiniButton::OnMotion**

void OnMotion(**const wxPoint&** *pos*)K

Responds to a mouse move event.

---

cbMiniButton::OnMotion
cbminibuttononmotion
browse00241
K cbMiniButton OnMotion
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbminibutton')")
K OnMotion

$#+K!**cbMiniButton::Refresh**

void Refresh()K

Refreshes the button.

---

cbMiniButton::Refresh
cbminibuttonrefresh
browse00242
K cbMiniButton Refresh
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbminibutton')")
K Refresh

$#+K!**cbMiniButton::Reset**

void Reset()K

Reset the button.

---

cbMiniButton::Reset
cbminibuttonreset
browse00243
K cbMiniButton Reset
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbminibutton')")
K Reset

$#+K!cbMiniButton::SetPos

void SetPos(**const wxPoint&** *pos*)K

Set the position of the button.

---

cbMiniButton::SetPos
cbminibuttonsetpos
browse00244
K cbMiniButton SetPos
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbminibutton')")
K SetPos

$#+K! cbMiniButton::WasClicked

bool WasClicked()K

Returns TRUE if the button was clicked.

---

cbMiniButton::WasClicked
cbminibuttonwasclicked
browse00245
K cbMiniButton WasClicked
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbminibutton')")
K WasClicked

$#+K!**cbMotionEvent::cbMotionEvent**

cbMotionEvent(**const wxPoint&** *pos*, **cbDockPane*** *pPane*)K

Constructor, taking mouse position and pane.

---

cbMotionEvent::cbMotionEvent
cbmotioneventcbmotionevent
browse00247
K cbMotionEvent cbMotionEvent
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbmotionevent')")
K cbMotionEvent

$#+K!**cbPaneDrawPlugin::cbPaneDrawPlugin**

cbPaneDrawPlugin(**wxFrameLayout*** *pPanel*, **int** *paneMask = wxALL_PANES*)K

Constructor taking frame layout pane and a pane mask.

cbPaneDrawPlugin()K

Default constructor.

---

cbPaneDrawPlugin::cbPaneDrawPlugin
cbpanedrawplugincbpanedrawplugin
browse00249
K cbPaneDrawPlugin cbPaneDrawPlugin
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbpanedrawplugin')")
K cbPaneDrawPlugin
K cbPaneDrawPlugin

$#+K!**cbPaneDrawPlugin::~cbPaneDrawPlugin**

~cbPaneDrawPlugin()K

Destructor.

---

cbPaneDrawPlugin::~cbPaneDrawPlugin
cbpanedrawplugindtor
browse00250
K cbPaneDrawPlugin ~cbPaneDrawPlugin
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbpanedrawplugin')")
K ~cbPaneDrawPlugin

$#+K!cbPaneDrawPlugin::Clone

cbPluginBase* Clone()K

Clone function, returning a new instance of this class.

---

cbPaneDrawPlugin::Clone
cbpanedrawpluginclone
browse00251
K cbPaneDrawPlugin Clone
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbpanedrawplugin')")
K Clone

$#+K!**cbPaneDrawPlugin::DrawBarInnerShadeRect**

void DrawBarInnerShadeRect(**cbBarInfo*** *pBar*, **wxDC&** *dc*)K

Internal helper: draws the inner bar shading.

---

cbPaneDrawPlugin::DrawBarInnerShadeRect
cbpanedrawplugindrawbarinnershaderect
browse00252
K cbPaneDrawPlugin DrawBarInnerShadeRect
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbpanedrawplugin')")
K DrawBarInnerShadeRect

$#+K! cbPaneDrawPlugin::DrawDraggedHandle

void DrawDraggedHandle(**const wxPoint&** *pos*, **cbDockPane&** *pane*)[K]

Internal helper: draws the dragged handle.

---

[c]bPaneDrawPlugin::DrawDraggedHandle
[c]bpanedrawplugindrawdraggedhandle
[b]rowse00253
[K] cbPaneDrawPlugin DrawDraggedHandle
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbpanedrawplugin')")
[K] DrawDraggedHandle

$#+K!cbPaneDrawPlugin::DrawLowerRowHandle

void DrawLowerRowHandle(**cbRowInfo*** *pRow*, **wxDC&** *dc*)[K]

Internal helper: draws the lower row handle.

---

[c]bPaneDrawPlugin::DrawLowerRowHandle
[c]bpanedrawplugindrawlowerrowhandle
[b]rowse00254
[K] cbPaneDrawPlugin DrawLowerRowHandle
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbpanedrawplugin')")
[K] DrawLowerRowHandle

$#+K!**cbPaneDrawPlugin::DrawLowerRowShades**

void DrawLowerRowShades(**cbRowInfo*** *pRow*, **wxDC&** *dc*, **int** *level*)[K]

Internal helper: draws the lower row shading.

---

[c]bPaneDrawPlugin::DrawLowerRowShades
[c]bpanedrawplugindrawlowerrowshades
[b]rowse00255
[K] cbPaneDrawPlugin DrawLowerRowShades
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbpanedrawplugin')")
[K] DrawLowerRowShades

$#+K!**cbPaneDrawPlugin::DrawPaneShade**

void DrawPaneShade(**wxDC&** *dc*, **int** *alignment*)K

Internal helper: draws the pane shading.

---

cbPaneDrawPlugin::DrawPaneShade
cbpanedrawplugindrawpaneshade
browse00256
K cbPaneDrawPlugin DrawPaneShade
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbpanedrawplugin')")
K DrawPaneShade

$#+K!**cbPaneDrawPlugin::DrawPaneShadeForRow**

void DrawPaneShadeForRow(**cbRowInfo*** *pRow*, **wxDC&** *dc*)[K]

Internal helper: draws the pane shading for a row.

---

[c]bPaneDrawPlugin::DrawPaneShadeForRow
[c]bpanedrawplugindrawpaneshadeforrow
[b]rowse00257
[K] cbPaneDrawPlugin DrawPaneShadeForRow
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbpanedrawplugin')")
[K] DrawPaneShadeForRow

$#+K!**cbPaneDrawPlugin::DrawShade**

void DrawShade(**int** *level*, **wxRect&** *rect*, **int** *alignment*, **wxDC&** *dc*)K

Internal helper: draws shading.

---

cbPaneDrawPlugin::DrawShade
cbpanedrawplugindrawshade
browse00258
K cbPaneDrawPlugin DrawShade
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbpanedrawplugin')")
K DrawShade

$#+K!**cbPaneDrawPlugin::DrawShade1**

void DrawShade1(**int** *level*, **wxRect&** *rect*, **int** *alignment*, **wxDC&** *dc*)[K]

Internal helper: draws shading.

---

[c]bPaneDrawPlugin::DrawShade1
[c]bpanedrawplugindrawshade1
[b]rowse00259
[K] cbPaneDrawPlugin DrawShade1
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbpanedrawplugin')")
[K] DrawShade1

$#+K!cbPaneDrawPlugin::DrawUpperRowHandle

void DrawUpperRowHandle(**cbRowInfo*** *pRow*, **wxDC&** *dc*)K

Internal helper: draws the upper row handle.

---

cbPaneDrawPlugin::DrawUpperRowHandle
cbpanedrawplugindrawupperrowhandle
browse00260
K cbPaneDrawPlugin DrawUpperRowHandle
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbpanedrawplugin')")
K DrawUpperRowHandle

$#+K!cbPaneDrawPlugin::DrawUpperRowShades

void DrawUpperRowShades(**cbRowInfo*** *pRow*, **wxDC&** *dc*, **int** *level*)[K]

Internal helper: draws the upper row shading.

---

[c]bPaneDrawPlugin::DrawUpperRowShades
[c]bpanedrawplugindrawupperrowshades
[b]rowse00261
[K] cbPaneDrawPlugin DrawUpperRowShades
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbpanedrawplugin')")
[K] DrawUpperRowShades

$#+K!cbPaneDrawPlugin::OnDrawBarDecorations

void OnDrawBarDecorations(**cbDrawBarDecorEvent&** *event*)K

Handler for draw bar decorations events.

---

cbPaneDrawPlugin::OnDrawBarDecorations
cbpanedrawpluginondrawbardecorations
browse00262
K cbPaneDrawPlugin OnDrawBarDecorations
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbpanedrawplugin')")
K OnDrawBarDecorations

$#+K!**cbPaneDrawPlugin::OnDrawBarHandles**

void OnDrawBarHandles(**cbDrawBarHandlesEvent&** *event*)K

Handler for draw bar handles events.

---

cbPaneDrawPlugin::OnDrawBarHandles
cbpanedrawpluginondrawbarhandles
browse00263
K cbPaneDrawPlugin OnDrawBarHandles
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbpanedrawplugin')")
K OnDrawBarHandles

$#+K!**cbPaneDrawPlugin::OnDrawPaneBackground**

void OnDrawPaneBackground(**cbDrawPaneBkGroundEvent&** *event*)K

Handler for draw pane background events.

---

cbPaneDrawPlugin::OnDrawPaneBackground
cbpanedrawpluginondrawpanebackground
browse00264
K cbPaneDrawPlugin OnDrawPaneBackground
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbpanedrawplugin')")
K OnDrawPaneBackground

$#+K!cbPaneDrawPlugin::OnDrawPaneDecorations

void OnDrawPaneDecorations(**cbDrawPaneDecorEvent&** *event*)[K]

Handler for draw pane decoration events.

---

[c]bPaneDrawPlugin::OnDrawPaneDecorations
[c]bpanedrawpluginondrawpanedecorations
[b]rowse00265
[K] cbPaneDrawPlugin OnDrawPaneDecorations
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbpanedrawplugin')")
[K] OnDrawPaneDecorations

$#+K!**cbPaneDrawPlugin::OnDrawRowBackground**

void OnDrawRowBackground(**cbDrawRowBkGroundEvent&** *event*)K

Handler for draw row background events.

---

cbPaneDrawPlugin::OnDrawRowBackground
cbpanedrawpluginondrawrowbackground
browse00266
K cbPaneDrawPlugin OnDrawRowBackground
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbpanedrawplugin')")
K OnDrawRowBackground

$#+K!cbPaneDrawPlugin::OnDrawRowDecorations

void OnDrawRowDecorations(**cbDrawRowDecorEvent&** *event*)K

Handler for draw row decoration events.

---

cbPaneDrawPlugin::OnDrawRowDecorations
cbpanedrawpluginondrawrowdecorations
browse00267
K cbPaneDrawPlugin OnDrawRowDecorations
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbpanedrawplugin')")
K OnDrawRowDecorations

$#+K!cbPaneDrawPlugin::OnDrawRowHandles

void OnDrawRowHandles(**cbDrawRowHandlesEvent&** *event*)K

Handler for draw row handles events.

---

cbPaneDrawPlugin::OnDrawRowHandles
cbpanedrawpluginondrawrowhandles
browse00268
K cbPaneDrawPlugin OnDrawRowHandles
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbpanedrawplugin')")
K OnDrawRowHandles

$#+K!cbPaneDrawPlugin::OnFinishDrawInArea

void OnFinishDrawInArea(**cbFinishDrawInAreaEvent&** *event*)[K]

Handler for finish draw in area events.

---

[c]bPaneDrawPlugin::OnFinishDrawInArea
[c]bpanedrawpluginonfinishdrawinarea
[b]rowse00269
[K] cbPaneDrawPlugin OnFinishDrawInArea
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbpanedrawplugin')")
[K] OnFinishDrawInArea

$#+K!cbPaneDrawPlugin::OnLButtonDown

void OnLButtonDown(**cbLeftDownEvent&** *event*)K

Handler for left mouse button down events.

---

cbPaneDrawPlugin::OnLButtonDown
cbpanedrawpluginonlbuttondown
browse00270
K cbPaneDrawPlugin OnLButtonDown
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbpanedrawplugin')")
K OnLButtonDown

$#+K!cbPaneDrawPlugin::OnLButtonUp

void OnLButtonUp(**cbLeftUpEvent&** *event*)[K]

Handler for left mouse button up events.

---

[c]bPaneDrawPlugin::OnLButtonUp
[c]bpanedrawpluginonlbuttonup
[b]rowse00271
[K] cbPaneDrawPlugin OnLButtonUp
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbpanedrawplugin')")
[K] OnLButtonUp

$#+K!cbPaneDrawPlugin::OnLDblClick

void OnLDblClick(cbLeftDClickEvent& *event*)K

Handler for left double-click mouse button down events.

---

cbPaneDrawPlugin::OnLDblClick
cbpanedrawpluginonldblclick
browse00272
K cbPaneDrawPlugin OnLDblClick
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbpanedrawplugin')")
K OnLDblClick

$#+K!cbPaneDrawPlugin::OnMouseMove

void OnMouseMove(**cbMotionEvent&** *event*)K

Handler for mouse move events.

---

cbPaneDrawPlugin::OnMouseMove
cbpanedrawpluginonmousemove
browse00273
K cbPaneDrawPlugin OnMouseMove
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbpanedrawplugin')")
K OnMouseMove

$#+K!cbPaneDrawPlugin::OnRButtonUp

void OnRButtonUp(**cbRightUpEvent&** *event*)[K]

Handler for right mouse button up events.

---

[c]bPaneDrawPlugin::OnRButtonUp
[c]bpanedrawpluginonrbuttonup
[b]rowse00274
[K] cbPaneDrawPlugin OnRButtonUp
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbpanedrawplugin')")
[K] OnRButtonUp

$#+K!cbPaneDrawPlugin::OnSizeBarWindow

void OnSizeBarWindow(**cbSizeBarWndEvent&** *event*)K

Handler for bar size events.

---

cbPaneDrawPlugin::OnSizeBarWindow
cbpanedrawpluginonsizebarwindow
browse00275
K cbPaneDrawPlugin OnSizeBarWindow
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbpanedrawplugin')")
K OnSizeBarWindow

$#+K!cbPaneDrawPlugin::OnStartDrawInArea

void OnStartDrawInArea(**cbStartDrawInAreaEvent&** *event*)K

Handler for start draw in area events.

---

cbPaneDrawPlugin::OnStartDrawInArea
cbpanedrawpluginonstartdrawinarea
browse00276
K cbPaneDrawPlugin OnStartDrawInArea
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbpanedrawplugin')")
K OnStartDrawInArea

$#+K!cbPaneDrawPlugin::SetDarkPixel

void SetDarkPixel(**int** *x*, **int** *y*, **wxDC&** *dc*)[K]

Internal helper: sets a dark pixel at the given location.

---

[c]bPaneDrawPlugin::SetDarkPixel
[c]bpanedrawpluginsetdarkpixel
[b]rowse00277
[K] cbPaneDrawPlugin SetDarkPixel
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbpanedrawplugin')")
[K] SetDarkPixel

$#+K!**cbPaneDrawPlugin::SetLightPixel**

void SetLightPixel(**int** *x*, **int** *y*, **wxDC&** *dc*)K

Internal helper: sets a light pixel at the given location.

---

cbPaneDrawPlugin::SetLightPixel
cbpanedrawpluginsetlightpixel
browse00278
K cbPaneDrawPlugin SetLightPixel
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbpanedrawplugin')")
K SetLightPixel

$#+K!**cbPluginBase::cbPluginBase**

cbPluginBase(**wxFrameLayout*** *pPanel*, **int** *paneMask* = *wxALL_PANES*)K

Constructor taking layout panel and a mask.

cbPluginBase()K

Default constructor.

---

cbPluginBase::cbPluginBase
cbpluginbasecbpluginbase
browse00280
K cbPluginBase cbPluginBase
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbpluginbase')")
K cbPluginBase
K cbPluginBase

$#+K!**cbPluginBase::~cbPluginBase**

~cbPluginBase()K

Destructor. Destroys the whole plugin chain of connected plugins.

---

cbPluginBase::~cbPluginBase
cbpluginbasedtor
browse00281
K cbPluginBase ~cbPluginBase
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbpluginbase')")
K ~cbPluginBase

$#+K!cbPluginBase::GetPaneMask

int GetPaneMask()K

Returns the pane mask.

---

cbPluginBase::GetPaneMask
cbpluginbasegetpanemask
browse00282
K cbPluginBase GetPaneMask
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbpluginbase')")
K GetPaneMask

$#+K!cbPluginBase::IsReady

bool IsReady()K

Returns TRUE if the plugin is ready to receive events.

---

cbPluginBase::IsReady
cbpluginbaseisready
browse00283
K cbPluginBase IsReady
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbpluginbase')")
K IsReady

$#+K!cbPluginBase::OnInitPlugin

void OnInitPlugin()K

Override this method to do plugin-specific initialization. At this point plugin is already attached to the frame layout, and pane masks are set.

---

cbPluginBase::OnInitPlugin
cbpluginbaseoninitplugin
browse00284
K cbPluginBase OnInitPlugin
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbpluginbase')")
K OnInitPlugin

$#+K!cbPluginBase::ProcessEvent

bool ProcessEvent(**wxEvent&** *event*)K

Overridden to determine whether the target pane specified in the event matches the pane mask of this plugin (specific plugins do not override this method).

---

cbPluginBase::ProcessEvent
cbpluginbaseprocessevent
browse00285
K cbPluginBase ProcessEvent
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbpluginbase')")
K ProcessEvent

$#+K!**cbPluginEvent::cbPluginEvent**

cbPluginEvent(**wxEventType** *eventType*, **cbDockPane*** *pPane*)[K]

Constructor, taking event type and pane.

---

[c]bPluginEvent::cbPluginEvent
[c]bplugineventcbplugineventbrowse00287
[K] cbPluginEvent cbPluginEvent
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbpluginevent')")
[K] cbPluginEvent

$#+KK!**cbPluginEvent::Clone**

wxEvent* Clone() **const**

Not used, but required.

---

cbPluginEvent::Clone
cbplugineventclone
browse00288
K cbPluginEvent Clone
K Clone
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbpluginevent')")

$#+K!**cbRemoveBarEvent::cbRemoveBarEvent**

cbRemoveBarEvent(**cbBarInfo*** *pBar*, **cbDockPane*** *pPane*)[K]

Constructor, taking bar information and pane.

---

[c]bRemoveBarEvent::cbRemoveBarEvent
[c]bremovebareventcbremovebarevent
[b]rowse00290
[K] cbRemoveBarEvent cbRemoveBarEvent
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbremovebarevent')")
[K] cbRemoveBarEvent

$#+K!cbResizeBarEvent::cbResizeBarEvent

cbResizeBarEvent(**cbBarInfo*** *pBar*, **cbRowInfo*** *pRow*, **cbDockPane*** *pPane*)[K]

Constructor, taking bar information, row information, and pane.

---

[c]bResizeBarEvent::cbResizeBarEvent
[c]bresizebareventcbresizebarevent
[b]rowse00292
[K] cbResizeBarEvent cbResizeBarEvent
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbresizebarevent')")
[K] cbResizeBarEvent

$#+K!**cbResizeRowEvent::cbResizeRowEvent**

cbResizeRowEvent(**cbRowInfo*** *pRow*, **int** *handleOfs*, **bool** *forUpperHandle*, **cbDockPane*** *pPane*)[K]

Constructor, taking row information, two parameters of currently unknown use, and pane.

---

[c]bResizeRowEvent::cbResizeRowEvent
[c]bresizeroweventcbresizerowevent
[b]rowse00294
[K] cbResizeRowEvent cbResizeRowEvent
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbresizerowevent')")
[K] cbResizeRowEvent

$#+K!cbRightDownEvent::cbRightDownEvent

cbRightDownEvent(**const wxPoint&** *pos*, **cbDockPane*** *pPane*)[K]

Constructor, taking mouse position and pane.

---

[c]bRightDownEvent::cbRightDownEvent
[c]brightdowneventcbrightdownevent
[b]rowse00296
[K] cbRightDownEvent cbRightDownEvent
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbrightdownevent')")
[K] cbRightDownEvent

$#+K!**cbRightUpEvent::cbRightUpEvent**

cbRightUpEvent(**const wxPoint&** *pos*, **cbDockPane*** *pPane*)K

Constructor, taking mouse position and pane.

---

cbRightUpEvent::cbRightUpEvent
cbrightupeventcbrightupevent
browse00298
K cbRightUpEvent cbRightUpEvent
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbrightupevent')")
K cbRightUpEvent

$#+K!cbRowDragPlugin::cbRowDragPlugin

cbRowDragPlugin(**wxFrameLayout*** *pLayout*, **int** *paneMask* = *wxALL_PANES*)[K]

Constructor, taking paren layout frame and pane mask.

cbRowDragPlugin()[K]

Default constructor.

---

[c]bRowDragPlugin::cbRowDragPlugin
[c]browdragplugincbrowdragplugin
[b]rowse00300
[K] cbRowDragPlugin cbRowDragPlugin
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbrowdragplugin')")
[K] cbRowDragPlugin
[K] cbRowDragPlugin

$#+K!**cbRowDragPlugin::~cbRowDragPlugin**

~cbRowDragPlugin()K

Destructor.

---

cbRowDragPlugin::~cbRowDragPlugin
cbrowdragplugindtor
browse00301
K cbRowDragPlugin ~cbRowDragPlugin
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbrowdragplugin')")
K ~cbRowDragPlugin

$#+K!**cbRowDragPlugin::CaptureDCArea**

wxBitmap* CaptureDCArea(**wxDC&** *dc*, **wxRect&** *area*)[K]

Helper for drag and drop.

---

[c]bRowDragPlugin::CaptureDCArea
[c]browdragplugincapturedcarea
[b]rowse00302
[K] cbRowDragPlugin CaptureDCArea
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbrowdragplugin')")
[K] CaptureDCArea

$#+K!cbRowDragPlugin::CheckPrevItemInFocus

void CheckPrevItemInFocus(**cbRowInfo*** *pRow*, **int** *iconIdx*)K

Helper for drag and drop.

---

cbRowDragPlugin::CheckPrevItemInFocus
cbrowdragplugincheckpreviteminfocus
browse00303
K cbRowDragPlugin CheckPrevItemInFocus
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbrowdragplugin')")
K CheckPrevItemInFocus

$#+K!cbRowDragPlugin::Clone

cbPluginBase* Clone()K

Clone function, returning a new instance of this class.

---

cbRowDragPlugin::Clone
cbrowdragpluginclone
browse00304
K cbRowDragPlugin Clone
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbrowdragplugin')")
K Clone

$#+K!**cbRowDragPlugin::CollapseRow**

void CollapseRow(**cbRowInfo*** *pRow*)K

Helper for drag and drop.

---

cbRowDragPlugin::CollapseRow
cbrowdragplugincollapserow
browse00305
K cbRowDragPlugin CollapseRow
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbrowdragplugin')")
K CollapseRow

$#+K!cbRowDragPlugin::Draw3DPattern

void Draw3DPattern(**wxRect&** *inRect*, **wxDC&** *dc*)K

Implements 'hard-coded metafile' for Netscape Navigator look.

---

cbRowDragPlugin::Draw3DPattern
cbrowdragplugindraw3dpattern
browse00306
K cbRowDragPlugin Draw3DPattern
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbrowdragplugin')")
K Draw3DPattern

$#+K!**cbRowDragPlugin::Draw3DRect**

void Draw3DRect(**wxRect&** *inRect*, **wxDC&** *dc*, **wxBrush&** *bkBrush*)K

Implements 'hard-coded metafile' for Netscape Navigator look.

---

cbRowDragPlugin::Draw3DRect
cbrowdragplugindraw3drect
browse00307
K cbRowDragPlugin Draw3DRect
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbrowdragplugin')")
K Draw3DRect

$#+K!cbRowDragPlugin::DrawCollapsedRowIcon

void DrawCollapsedRowIcon(**int** *index*, **wxDC&** *dc*, **bool** *isHighlighted*)K

Draws collapsed row icon (appearance-dependent).

---

cbRowDragPlugin::DrawCollapsedRowIcon
cbrowdragplugindrawcollapsedrowicon
browse00308
K cbRowDragPlugin DrawCollapsedRowIcon
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbrowdragplugin')")
K DrawCollapsedRowIcon

$#+K!cbRowDragPlugin::DrawCollapsedRowsBorder

void DrawCollapsedRowsBorder(**wxDC&** *dc*)[K]

Draws collapsed rows border (appearance-dependent).

---

[c]bRowDragPlugin::DrawCollapsedRowsBorder
[c]browdragplugindrawcollapsedrowsborder
[b]rowse00309
[K] cbRowDragPlugin DrawCollapsedRowsBorder
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbrowdragplugin')")
[K] DrawCollapsedRowsBorder

$#+K!**cbRowDragPlugin::DrawEmptyRow**

void DrawEmptyRow(**wxDC&** *dc*, **wxRect&** *rowBounds*)K

Draws empty row (appearance-dependent).

---

cbRowDragPlugin::DrawEmptyRow
cbrowdragplugindrawemptyrow
browse00310
K cbRowDragPlugin DrawEmptyRow
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbrowdragplugin')")
K DrawEmptyRow

$#+K!**cbRowDragPlugin::DrawOrtoRomb**

void DrawOrtoRomb(**wxRect&** *inRect*, **wxDC&** *dc*, **wxBrush&** *bkBrush*)K

Implements 'hard-coded metafile' for Netscape Navigator look.

---

cbRowDragPlugin::DrawOrtoRomb
cbrowdragplugindrawortoromb
browse00311
K cbRowDragPlugin DrawOrtoRomb
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbrowdragplugin')")
K DrawOrtoRomb

$#+K!**cbRowDragPlugin::DrawRectShade**

void DrawRectShade(**wxRect&** *inRect*, **wxDC&** *dc*, **int** *level*, **wxPen&** *upperPen*, **wxPen&** *lowerPen*)K

Implements 'hard-coded metafile' for Netscape Navigator look.

---

cbRowDragPlugin::DrawRectShade
cbrowdragplugindrawrectshade
browse00312
K cbRowDragPlugin DrawRectShade
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbrowdragplugin')")
K DrawRectShade

$#+K!**cbRowDragPlugin::DrawRomb**

void DrawRomb(**wxRect&** *inRect*, **wxDC&** *dc*, **wxBrush&** *bkBrush*)K

Implements 'hard-coded metafile' for Netscape Navigator look.

---

cbRowDragPlugin::DrawRomb
cbrowdragplugindrawromb
browse00313
K cbRowDragPlugin DrawRomb
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbrowdragplugin')")
K DrawRomb

$#+K!**cbRowDragPlugin::DrawRombShades**

void DrawRombShades(**wxPoint&** *p1*, **wxPoint&** *p2*, **wxPoint&** *p3*, **wxPoint&** *p4*, **wxDC&** *dc*)K

Implements 'hard-coded metafile' for Netscape Navigator look.

---

cbRowDragPlugin::DrawRombShades
cbrowdragplugindrawrombshades
browse00314
K cbRowDragPlugin DrawRombShades
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbrowdragplugin')")
K DrawRombShades

$#+K!**cbRowDragPlugin::DrawRowDragHint**

void DrawRowDragHint(**cbRowInfo*** *pRow*, **wxDC&** *dc*, **bool** *isHighlighted*)K

Draws row drag hint (appearance-dependent).

---

cbRowDragPlugin::DrawRowDragHint
cbrowdragplugindrawrowdraghint
browse00315
K cbRowDragPlugin DrawRowDragHint
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbrowdragplugin')")
K DrawRowDragHint

$#+K!cbRowDragPlugin::DrawRowsDragHintsBorder

void DrawRowsDragHintsBorder(**wxDC&** *dc*)[K]

Draws rows drag hints border (appearance-dependent).

---

[c]bRowDragPlugin::DrawRowsDragHintsBorder
[c]browdragplugindrawrowsdraghintsborder
[b]rowse00316
[K] cbRowDragPlugin DrawRowsDragHintsBorder
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbrowdragplugin')")
[K] DrawRowsDragHintsBorder

$#+K!**cbRowDragPlugin::DrawTrianDown**

void DrawTrianDown(**wxRect&** *inRect*, **wxDC&** *dc*)[K]

Implements 'hard-coded metafile' for Netscape Navigator look.

---

[c]bRowDragPlugin::DrawTrianDown
[c]browdragplugindrawtriandown
[b]rowse00317
[K] cbRowDragPlugin DrawTrianDown
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbrowdragplugin')")
[K] DrawTrianDown

$#+K!**cbRowDragPlugin::DrawTrianRight**

void DrawTrianRight(**wxRect&** *inRect*, **wxDC&** *dc*)K

Implements 'hard-coded metafile' for Netscape Navigator look.

---

cbRowDragPlugin::DrawTrianRight
cbrowdragplugindrawtrianright
browse00318
K cbRowDragPlugin DrawTrianRight
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbrowdragplugin')")
K DrawTrianRight

$#+K!**cbRowDragPlugin::DrawTrianUp**

void DrawTrianUp(**wxRect&** *inRect*, **wxDC&** *dc*)K

Implements 'hard-coded metafile' for Netscape Navigator look.

---

cbRowDragPlugin::DrawTrianUp
cbrowdragplugindrawtrianup
browse00319
K cbRowDragPlugin DrawTrianUp
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbrowdragplugin')")
K DrawTrianUp

$#+K!**cbRowDragPlugin::ExpandRow**

void ExpandRow(**int** *collapsedIconIdx*)[K]

Helper for drag and drop.

---

[c]bRowDragPlugin::ExpandRow
[c]browdragpluginexpandrow
[b]rowse00320
[K] cbRowDragPlugin ExpandRow
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbrowdragplugin')")
[K] ExpandRow

$#+K!cbRowDragPlugin::FinishOnScreenDraw

void FinishOnScreenDraw()K

Helper for drag and drop.

---

cbRowDragPlugin::FinishOnScreenDraw
cbrowdragpluginfinishonscreendraw
browse00321
K cbRowDragPlugin FinishOnScreenDraw
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbrowdragplugin')")
K FinishOnScreenDraw

$#+K!cbRowDragPlugin::GetCollapsedIconsPos

int GetCollapsedIconsPos()K

Helper for drag and drop.

---

cbRowDragPlugin::GetCollapsedIconsPos
cbrowdragplugingetcollapsediconspos
browse00322
K cbRowDragPlugin GetCollapsedIconsPos
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbrowdragplugin')")
K GetCollapsedIconsPos

$#+K!cbRowDragPlugin::GetCollapsedInconRect

void GetCollapsedInconRect(**int** *iconIdx*, **wxRect&** *rect*)K

Helper for drag and drop.

---

cbRowDragPlugin::GetCollapsedInconRect
cbrowdragplugingetcollapsedinconrect
browse00323
K cbRowDragPlugin GetCollapsedInconRect
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbrowdragplugin')")
K GetCollapsedInconRect

$#+K!cbRowDragPlugin::GetCollapsedRowIconHeight

int GetCollapsedRowIconHeight()K

Gets the collapsed row icon height.

---

cbRowDragPlugin::GetCollapsedRowIconHeight
cbrowdragplugingetcollapsedrowiconheight
browse00324
K cbRowDragPlugin GetCollapsedRowIconHeight
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbrowdragplugin')")
K GetCollapsedRowIconHeight

$#+K!**cbRowDragPlugin::GetFirstRow**

cbRowInfo* GetFirstRow()K

Helper for drag and drop.

---

cbRowDragPlugin::GetFirstRow
cbrowdragplugingetfirstrow
browse00325
K cbRowDragPlugin GetFirstRow
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbrowdragplugin')")
K GetFirstRow

$#+K!**cbRowDragPlugin::GetHRowsCountForPane**

int GetHRowsCountForPane(**cbDockPane*** *pPane*)K

Helper for drag and drop.

---

cbRowDragPlugin::GetHRowsCountForPane
cbrowdragplugingethrowscountforpane
browse00326
K cbRowDragPlugin GetHRowsCountForPane
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbrowdragplugin')")
K GetHRowsCountForPane

$#+K!**cbRowDragPlugin::GetRowDragHintWidth**

int GetRowDragHintWidth()K

Gets the row drag hint width.

---

cbRowDragPlugin::GetRowDragHintWidth
cbrowdragplugingetrowdraghintwidth
browse00327
K cbRowDragPlugin GetRowDragHintWidth
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbrowdragplugin')")
K GetRowDragHintWidth

$#+K!**cbRowDragPlugin::GetRowHintRect**

void GetRowHintRect(**cbRowInfo*** *pRow*, **wxRect&** *rect*)K

Helper for drag and drop.

---

cbRowDragPlugin::GetRowHintRect
cbrowdragplugingetrowhintrect
browse00328
K cbRowDragPlugin GetRowHintRect
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbrowdragplugin')")
K GetRowHintRect

$#+K!cbRowDragPlugin::HitTestCollapsedRowIcon

bool HitTestCollapsedRowIcon(**int** *iconIdx*, **const wxPoint&** *pos*)[K]

Test for the collapsed row icon position.

---

[c]bRowDragPlugin::HitTestCollapsedRowIcon
[c]browdragpluginhittestcollapsedrowicon
[b]rowse00329
[K] cbRowDragPlugin HitTestCollapsedRowIcon
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbrowdragplugin')")
[K] HitTestCollapsedRowIcon

$#+K!cbRowDragPlugin::HitTestRowDragHint

bool HitTestRowDragHint(**cbRowInfo*** *pRow*, **const wxPoint&** *pos*)K

Test for the row drag hint position.

---

cbRowDragPlugin::HitTestRowDragHint
cbrowdragpluginhittestrowdraghint
browse00330
K cbRowDragPlugin HitTestRowDragHint
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbrowdragplugin')")
K HitTestRowDragHint

$#+K!cbRowDragPlugin::InsertDraggedRowBefore

void InsertDraggedRowBefore(**cbRowInfo*** *pBeforeRow*)[K]

Helper for drag and drop.

---

[c]bRowDragPlugin::InsertDraggedRowBefore
[c]browdragplugininsertdraggedrowbefore
[b]rowse00331
[K] cbRowDragPlugin InsertDraggedRowBefore
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbrowdragplugin')")
[K] InsertDraggedRowBefore

$#+K!cbRowDragPlugin::ItemIsInFocus

bool ItemIsInFocus()K

Helper for drag and drop.

---

cbRowDragPlugin::ItemIsInFocus
cbrowdragpluginitemisinfocus
browse00332
K cbRowDragPlugin ItemIsInFocus
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbrowdragplugin')")
K ItemIsInFocus

$#+K!cbRowDragPlugin::OnDrawPaneBackground

void OnDrawPaneBackground(**cbDrawPaneDecorEvent&** *event*)K

Handles pane drawing plugin events (appearance-independent logic).

---

cbRowDragPlugin::OnDrawPaneBackground
cbrowdragpluginondrawpanebackground
browse00333
K cbRowDragPlugin OnDrawPaneBackground
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbrowdragplugin')")
K OnDrawPaneBackground

$#+K!**cbRowDragPlugin::OnInitPlugin**

void OnInitPlugin()K

Called to initialize this plugin.

---

cbRowDragPlugin::OnInitPlugin
cbrowdragpluginoninitplugin
browse00334
K cbRowDragPlugin OnInitPlugin
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbrowdragplugin')")
K OnInitPlugin

$#+K!**cbRowDragPlugin::OnLButtonDown**

void OnLButtonDown(**cbLeftDownEvent&** *event*)K

Handles left button down plugin events (appearance-independent logic).

---

cbRowDragPlugin::OnLButtonDown
cbrowdragpluginonlbuttondown
browse00335
K cbRowDragPlugin OnLButtonDown
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbrowdragplugin')")
K OnLButtonDown

$#+K!cbRowDragPlugin::OnLButtonUp

void OnLButtonUp(**cbLeftUpEvent&** *event*)K

Handles left button up plugin events (appearance-independent logic).

---

cbRowDragPlugin::OnLButtonUp
cbrowdragpluginonlbuttonup
browse00336
K cbRowDragPlugin OnLButtonUp
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbrowdragplugin')")
K OnLButtonUp

$#+K!cbRowDragPlugin::OnMouseMove

void OnMouseMove(**cbMotionEvent&** *event*)K

Handles mouse move plugin events (appearance-independent logic).

---

cbRowDragPlugin::OnMouseMove
cbrowdragpluginonmousemove
browse00337
K cbRowDragPlugin OnMouseMove
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbrowdragplugin')")
K OnMouseMove

$#+K!**cbRowDragPlugin::PrepareForRowDrag**

void PrepareForRowDrag()K

Helper for drag and drop.

---

cbRowDragPlugin::PrepareForRowDrag
cbrowdragpluginprepareforrowdrag
browse00338
K cbRowDragPlugin PrepareForRowDrag
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbrowdragplugin')")
K PrepareForRowDrag

$#+K!**cbRowDragPlugin::SetMouseCapture**

void SetMouseCapture(**bool** *captureOn*)[K]

Helper for drag and drop.

---

[c]bRowDragPlugin::SetMouseCapture
[c]browdragpluginsetmousecapture
[b]rowse00339
[K] cbRowDragPlugin SetMouseCapture
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbrowdragplugin')")
[K] SetMouseCapture

$#+K!cbRowDragPlugin::SetPaneMargins

void SetPaneMargins()K

Sets the pane margins.

---

cbRowDragPlugin::SetPaneMargins
cbrowdragpluginsetpanemargins
browse00340
K cbRowDragPlugin SetPaneMargins
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbrowdragplugin')")
K SetPaneMargins

$#+K!cbRowDragPlugin::ShowDraggedRow

void ShowDraggedRow(**int** *offset*)K

Helper for drag and drop.

---

cbRowDragPlugin::ShowDraggedRow
cbrowdragpluginshowdraggedrow
browse00341
K cbRowDragPlugin ShowDraggedRow
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbrowdragplugin')")
K ShowDraggedRow

$#+K!**cbRowDragPlugin::ShowPaneImage**

void ShowPaneImage()K

Helper for drag and drop.

---

cbRowDragPlugin::ShowPaneImage
cbrowdragpluginshowpaneimage
browse00342
K cbRowDragPlugin ShowPaneImage
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbrowdragplugin')")
K ShowPaneImage

$#+K!cbRowDragPlugin::UnhighlightItemInFocus

void UnhighlightItemInFocus()K

Helper for drag and drop.

---

cbRowDragPlugin::UnhighlightItemInFocus
cbrowdragpluginunhighlightiteminfocus
browse00343
K cbRowDragPlugin UnhighlightItemInFocus
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbrowdragplugin')")
K UnhighlightItemInFocus

$#+K!**cbRowInfo::cbRowInfo**

cbRowInfo()[K]

Constructor.

---

[c]bRowInfo::cbRowInfo
[c]browinfocbrowinfo
[b]rowse00345
[K] cbRowInfo cbRowInfo
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbrowinfo')")
[K] cbRowInfo

$#+K!**cbRowInfo::~cbRowInfo**

~cbRowInfo()K

Destructor.

---

cbRowInfo::~cbRowInfo
cbrowinfodtor
browse00346
K cbRowInfo ~cbRowInfo
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbrowinfo')")
K ~cbRowInfo

$#+K!**cbRowInfo::GetFirstBar**

cbBarInfo* GetFirstBar()K

Returns the first bar.

---

cbRowInfo::GetFirstBar
cbrowinfogetfirstbar
browse00347
K cbRowInfo GetFirstBar
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbrowinfo')")
K GetFirstBar

$#+K!**cbRowLayoutPlugin::cbRowLayoutPlugin**

cbRowLayoutPlugin(**wxFrameLayout*** *pPanel*, **int** *paneMask* = *wxALL_PANES*)K

Constructor taking frame layout pane and pane mask.

cbRowLayoutPlugin()K

Default constructor.

---

cbRowLayoutPlugin::cbRowLayoutPlugin
cbrowlayoutplugincbrowlayoutplugin
browse00349
K cbRowLayoutPlugin cbRowLayoutPlugin
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbrowlayoutplugin')")
K cbRowLayoutPlugin
K cbRowLayoutPlugin

$#+K!cbRowLayoutPlugin::AdjustLengthOfInserted

void AdjustLengthOfInserted(**cbRowInfo*** *pRow*, **cbBarInfo*** *pTheBar*)[K]

Internal helper relating to not-fixed-bars layout.

---

[c]bRowLayoutPlugin::AdjustLengthOfInserted
[c]browlayoutpluginadjustlengthofinserted
[b]rowse00350
[K] cbRowLayoutPlugin AdjustLengthOfInserted
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbrowlayoutplugin')")
[K] AdjustLengthOfInserted

$#+K!cbRowLayoutPlugin::ApplyLengthRatios

void ApplyLengthRatios(**cbRowInfo*** *pRow*)[K]

Internal helper relating to not-fixed-bars layout.

---

[c]bRowLayoutPlugin::ApplyLengthRatios
[c]browlayoutpluginapplylengthratios
[b]rowse00351
[K] cbRowLayoutPlugin ApplyLengthRatios
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbrowlayoutplugin')")
[K] ApplyLengthRatios

$#+K!**cbRowLayoutPlugin::CalcRowHeight**

int CalcRowHeight(**cbRowInfo&** *row*)K

Row layout helper simulating bar 'friction'.

---

cbRowLayoutPlugin::CalcRowHeight
cbrowlayoutplugincalcrowheight
browse00352
K cbRowLayoutPlugin CalcRowHeight
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbrowlayoutplugin')")
K CalcRowHeight

$#+K!cbRowLayoutPlugin::CheckIfAtTheBoundary

void CheckIfAtTheBoundary(**cbBarInfo*** *pTheBar*, **cbRowInfo&** *rowInfo*)[K]

Internal helper relating to not-fixed-bars layout.

---

[c]bRowLayoutPlugin::CheckIfAtTheBoundary
[c]browlayoutplugincheckifattheboundary
[b]rowse00353
[K] cbRowLayoutPlugin CheckIfAtTheBoundary
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbrowlayoutplugin')")
[K] CheckIfAtTheBoundary

$#+K!cbRowLayoutPlugin::DetectBarHandles

void DetectBarHandles(**cbRowInfo*** *pRow*)[K]

Internal helper relating to not-fixed-bars layout.

---

[c]bRowLayoutPlugin::DetectBarHandles
[c]browlayoutplugindetectbarhandles
[b]rowse00354
[K] cbRowLayoutPlugin DetectBarHandles
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbrowlayoutplugin')")
[K] DetectBarHandles

$#+K!**cbRowLayoutPlugin::DoInsertBar**

void DoInsertBar(**cbBarInfo*** *pTheBar*, **cbRowInfo&** *row*)K

Insert the bar before the given row.

---

cbRowLayoutPlugin::DoInsertBar
cbrowlayoutplugindoinsertbar
browse00355
K cbRowLayoutPlugin DoInsertBar
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbrowlayoutplugin')")
K DoInsertBar

$#+K!cbRowLayoutPlugin::ExpandNotFixedBars

void ExpandNotFixedBars(**cbRowInfo*** *pRow*)[K]

Internal helper relating to not-fixed-bars layout.

---

[c]bRowLayoutPlugin::ExpandNotFixedBars
[c]browlayoutpluginexpandnotfixedbars
[b]rowse00356
[K] cbRowLayoutPlugin ExpandNotFixedBars
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbrowlayoutplugin')")
[K] ExpandNotFixedBars

$#+K!cbRowLayoutPlugin::FitBarsToRange

void FitBarsToRange(**int** *from*, **int** *till*, **cbBarInfo*** *pTheBar*, **cbRowInfo*** *pRow*)[K]

Internal helper relating to not-fixed-bars layout.

---

[c]bRowLayoutPlugin::FitBarsToRange
[c]browlayoutpluginfitbarstorange
[b]rowse00357
[K] cbRowLayoutPlugin FitBarsToRange
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbrowlayoutplugin')")
[K] FitBarsToRange

$#+K!cbRowLayoutPlugin::GetRowFreeSpace

int GetRowFreeSpace(**cbRowInfo*** *pRow*)K

Internal helper relating to not-fixed-bars layout.

---

cbRowLayoutPlugin::GetRowFreeSpace
cbrowlayoutplugingetrowfreespace
browse00358
K cbRowLayoutPlugin GetRowFreeSpace
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbrowlayoutplugin')")
K GetRowFreeSpace

$#+K!cbRowLayoutPlugin::InsertBefore

void InsertBefore(**cbBarInfo*** *pBeforeBar*, **cbBarInfo*** *pTheBar*, **cbRowInfo&** *row*)K

Insert the bar before the given row.

---

cbRowLayoutPlugin::InsertBefore
cbrowlayoutplugininsertbefore
browse00359
K cbRowLayoutPlugin InsertBefore
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbrowlayoutplugin')")
K InsertBefore

$#+K!**cbRowLayoutPlugin::LayoutItemsVertically**

void LayoutItemsVertically(**cbRowInfo&** *row*)[K]

Row layout helper simulating bar 'friction'.

---

[c]bRowLayoutPlugin::LayoutItemsVertically
[c]browlayoutpluginlayoutitemsvertically
[b]rowse00360
[K] cbRowLayoutPlugin LayoutItemsVertically
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbrowlayoutplugin')")
[K] LayoutItemsVertically

$#+K!**cbRowLayoutPlugin::MinimzeNotFixedBars**

void MinimzeNotFixedBars(**cbRowInfo*** *pRow*, **cbBarInfo*** *pBarToPreserve*)[K]

Internal helper relating to not-fixed-bars layout.

---

[c]bRowLayoutPlugin::MinimzeNotFixedBars
[c]browlayoutpluginminimzenotfixedbars
[b]rowse00361
[K] cbRowLayoutPlugin MinimzeNotFixedBars
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbrowlayoutplugin')")
[K] MinimzeNotFixedBars

$#+K!**cbRowLayoutPlugin::OnInsertBar**

void OnInsertBar(**cbInsertBarEvent&** *event*)K

Responds to bar insertion event.

---

cbRowLayoutPlugin::OnInsertBar
cbrowlayoutpluginoninsertbar
browse00362
K cbRowLayoutPlugin OnInsertBar
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbrowlayoutplugin')")
K OnInsertBar

$#+K!**cbRowLayoutPlugin::OnLayoutRow**

void OnLayoutRow(**cbLayoutRowEvent&** *event*)K

Responds to row layout event.

---

cbRowLayoutPlugin::OnLayoutRow
cbrowlayoutpluginonlayoutrow
browse00363
K cbRowLayoutPlugin OnLayoutRow
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbrowlayoutplugin')")
K OnLayoutRow

$#+K!cbRowLayoutPlugin::OnLayoutRows

void OnLayoutRows(**cbLayoutRowsEvent&** *event*)K

Responds to rows layout event.

---

cbRowLayoutPlugin::OnLayoutRows
cbrowlayoutpluginonlayoutrows
browse00364
K cbRowLayoutPlugin OnLayoutRows
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbrowlayoutplugin')")
K OnLayoutRows

$#+K!cbRowLayoutPlugin::OnRemoveBar

void OnRemoveBar(**cbRemoveBarEvent&** *event*)K

Responds to bar removal event.

---

cbRowLayoutPlugin::OnRemoveBar
cbrowlayoutpluginonremovebar
browse00365
K cbRowLayoutPlugin OnRemoveBar
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbrowlayoutplugin')")
K OnRemoveBar

$#+K!cbRowLayoutPlugin::OnResizeRow

void OnResizeRow(**cbResizeRowEvent&** *event*)K

Responds to row resize event.

---

cbRowLayoutPlugin::OnResizeRow
cbrowlayoutpluginonresizerow
browse00366
K cbRowLayoutPlugin OnResizeRow
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbrowlayoutplugin')")
K OnResizeRow

$#+K!cbRowLayoutPlugin::RecalcLengthRatios

void RecalcLengthRatios(**cbRowInfo*** *pRow*)[K]

Internal helper relating to not-fixed-bars layout.

---

[c]bRowLayoutPlugin::RecalcLengthRatios
[c]browlayoutpluginrecalclengthratios
[b]rowse00367
[K] cbRowLayoutPlugin RecalcLengthRatios
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbrowlayoutplugin')")
[K] RecalcLengthRatios

$#+K!cbRowLayoutPlugin::RelayoutNotFixedBarsAround

void RelayoutNotFixedBarsAround(**cbBarInfo*** *pTheBar*, **cbRowInfo*** *pRow*)K

Internal helper relating to not-fixed-bars layout.

---

cbRowLayoutPlugin::RelayoutNotFixedBarsAround
cbrowlayoutpluginrelayoutnotfixedbarsaround
browse00368
K cbRowLayoutPlugin RelayoutNotFixedBarsAround
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbrowlayoutplugin')")
K RelayoutNotFixedBarsAround

$#+K!cbRowLayoutPlugin::ShiftLeftTrashold

void ShiftLeftTrashold(**cbBarInfo*** *pTheBar*, **cbRowInfo&** *row*)[K]

Row layout helper simulating bar 'friction'.

---

[c]bRowLayoutPlugin::ShiftLeftTrashold
[c]browlayoutpluginshiftlefttrashold
[b]rowse00369
[K] cbRowLayoutPlugin ShiftLeftTrashold
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbrowlayoutplugin')")
[K] ShiftLeftTrashold

$#+K!cbRowLayoutPlugin::ShiftRightTrashold

void ShiftRightTrashold(**cbBarInfo*** *pTheBar*, **cbRowInfo&** *row*)K

Row layout helper simulating bar 'friction'.

---

cbRowLayoutPlugin::ShiftRightTrashold
cbrowlayoutpluginshiftrighttrashold
browse00370
K cbRowLayoutPlugin ShiftRightTrashold
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbrowlayoutplugin')")
K ShiftRightTrashold

$#+K!**cbRowLayoutPlugin::SlideLeftSideBars**

void SlideLeftSideBars(**cbBarInfo*** *pTheBar*)K

Row layout helper simulating bar 'friction'.

---

cbRowLayoutPlugin::SlideLeftSideBars
cbrowlayoutpluginslideleftsidebars
browse00371
K cbRowLayoutPlugin SlideLeftSideBars
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbrowlayoutplugin')")
K SlideLeftSideBars

$#+K!cbRowLayoutPlugin::SlideRightSideBars

void SlideRightSideBars(**cbBarInfo*** *pTheBar*)K

Row layout helper simulating bar 'friction'.

---

cbRowLayoutPlugin::SlideRightSideBars
cbrowlayoutpluginsliderightsidebars
browse00372
K cbRowLayoutPlugin SlideRightSideBars
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbrowlayoutplugin')")
K SlideRightSideBars

$#+K!cbRowLayoutPlugin::StickRightSideBars

void StickRightSideBars(**cbBarInfo*** *pToBar*)K

Row layout helper simulating bar 'friction'.

---

cbRowLayoutPlugin::StickRightSideBars
cbrowlayoutpluginstickrightsidebars
browse00373
K cbRowLayoutPlugin StickRightSideBars
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbrowlayoutplugin')")
K StickRightSideBars

$#+K!**cbSimpleCustomizationPlugin::cbSimpleCustomizationPlugin**

cbSimpleCustomizationPlugin(**wxFrameLayout*** *pPanel*, **int** *paneMask* = *wxALL_PANES*)K

Constructor, taking parent pane and a pane mask flag.

cbSimpleCustomizationPlugin()K

Default constructor.

---

cbSimpleCustomizationPlugin::cbSimpleCustomizationPlugin
cbsimplecustomizationplugincbsimplecustomizationplugin
browse00375
K cbSimpleCustomizationPlugin cbSimpleCustomizationPlugin
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbsimplecustomizationplugin')")
K cbSimpleCustomizationPlugin
K cbSimpleCustomizationPlugin

$#+K!cbSimpleCustomizationPlugin::OnCustomizeBar

void OnCustomizeBar(**cbCustomizeBarEvent&** *event*)[K]

Plugin event handler for cbCustomizeBarEvent.

---

[c]bSimpleCustomizationPlugin::OnCustomizeBar
[c]bsimplecustomizationpluginoncustomizebar
[b]rowse00376
[K] cbSimpleCustomizationPlugin OnCustomizeBar
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbsimplecustomizationplugin')")
[K] OnCustomizeBar

$#+K!cbSimpleCustomizationPlugin::OnCustomizeLayout

void OnCustomizeLayout(**cbCustomizeLayoutEvent&** *event*)K

Plugin event handler for cbCustomizeLayoutEvent.

---

cbSimpleCustomizationPlugin::OnCustomizeLayout
cbsimplecustomizationpluginoncustomizelayout
browse00377
K cbSimpleCustomizationPlugin OnCustomizeLayout
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbsimplecustomizationplugin')")
K OnCustomizeLayout

$#+K!**cbSimpleCustomizationPlugin::OnMenuItemSelected**

void OnMenuItemSelected(**wxCommandEvent&** *event*)K

Menu event handler.

---

cbSimpleCustomizationPlugin::OnMenuItemSelected
cbsimplecustomizationpluginonmenuitemselected
browse00378
K cbSimpleCustomizationPlugin OnMenuItemSelected
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbsimplecustomizationplugin')")
K OnMenuItemSelected

$#+K!**cbSimpleUpdatesMgr::cbSimpleUpdatesMgr**

cbSimpleUpdatesMgr()K

Default constructor.

cbSimpleUpdatesMgr(**wxFrameLayout*** *pPanel*)K

Constructor taking frame layout panel.

---

cbSimpleUpdatesMgr::cbSimpleUpdatesMgr
cbsimpleupdatesmgrcbsimpleupdatesmgr
browse00380
K cbSimpleUpdatesMgr cbSimpleUpdatesMgr
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbsimpleupdatesmgr')")
K cbSimpleUpdatesMgr
K cbSimpleUpdatesMgr

$#+K!cbSimpleUpdatesMgr::OnBarWillChange

void OnBarWillChange(**cbBarInfo*** *pBar*, **cbRowInfo*** *pInRow*, **cbDockPane*** *pInPane*)[K]

Notification received from Frame Layout in the order in which they would usually be invoked.

---

[c]bSimpleUpdatesMgr::OnBarWillChange
[c]bsimpleupdatesmgronbarwillchange
[b]rowse00381
[K] cbSimpleUpdatesMgr OnBarWillChange
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbsimpleupdatesmgr')")
[K] OnBarWillChange

$#+K!cbSimpleUpdatesMgr::OnFinishChanges

void OnFinishChanges()K

Notification received from Frame Layout in the order in which they would usually be invoked.

---

cbSimpleUpdatesMgr::OnFinishChanges
cbsimpleupdatesmgronfinishchanges
browse00382
K cbSimpleUpdatesMgr OnFinishChanges
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbsimpleupdatesmgr')")
K OnFinishChanges

$#+K!cbSimpleUpdatesMgr::OnPaneMarginsWillChange

void OnPaneMarginsWillChange(**cbDockPane*** *pPane*)[K]

Notification received from Frame Layout in the order in which they would usually be invoked.

---

[c]bSimpleUpdatesMgr::OnPaneMarginsWillChange
[c]bsimpleupdatesmgronpanemarginswillchange
[b]rowse00383
[K] cbSimpleUpdatesMgr OnPaneMarginsWillChange
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbsimpleupdatesmgr')")
[K] OnPaneMarginsWillChange

$#+K!cbSimpleUpdatesMgr::OnPaneWillChange

void OnPaneWillChange(**cbDockPane*** *pPane*)[K]

Notification received from Frame Layout in the order in which they would usually be invoked.

---

[c]bSimpleUpdatesMgr::OnPaneWillChange
[c]bsimpleupdatesmgronpanewillchange
[b]rowse00384
[K] cbSimpleUpdatesMgr OnPaneWillChange
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbsimpleupdatesmgr')")
[K] OnPaneWillChange

$#+K!cbSimpleUpdatesMgr::OnRowWillChange

void OnRowWillChange(**cbRowInfo*** *pRow*, **cbDockPane*** *pInPane*)[K]

Notification received from Frame Layout in the order in which they would usually be invoked.

---

[c]bSimpleUpdatesMgr::OnRowWillChange
[c]bsimpleupdatesmgronrowwillchange
[b]rowse00385
[K] cbSimpleUpdatesMgr OnRowWillChange
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbsimpleupdatesmgr')")
[K] OnRowWillChange

$#+K!cbSimpleUpdatesMgr::OnStartChanges

void OnStartChanges()K

Notification received from Frame Layout in the order in which they would usually be invoked.

---

cbSimpleUpdatesMgr::OnStartChanges
cbsimpleupdatesmgronstartchanges
browse00386
K cbSimpleUpdatesMgr OnStartChanges
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbsimpleupdatesmgr')")
K OnStartChanges

$#+K!cbSimpleUpdatesMgr::UpdateNow

void UpdateNow()[K]

Refreshes the parts of the frame layoutthat need an update.

---

[c]bSimpleUpdatesMgr::UpdateNow
[c]bsimpleupdatesmgrupdatenow
[b]rowse00387
[K] cbSimpleUpdatesMgr UpdateNow
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbsimpleupdatesmgr')")
[K] UpdateNow

$#+K!**cbSimpleUpdatesMgr::WasChanged**

bool WasChanged(**cbUpdateMgrData&** *data*, **wxRect&** *currentBounds*)K

Helper function.

---

cbSimpleUpdatesMgr::WasChanged
cbsimpleupdatesmgrwaschanged
browse00388
K cbSimpleUpdatesMgr WasChanged
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbsimpleupdatesmgr')")
K WasChanged

$#+K!**cbSizeBarWndEvent::cbSizeBarWndEvent**

cbSizeBarWndEvent(**cbBarInfo*** *pBar*, **cbDockPane*** *pPane*)K

Constructor, taking bar information and pane.

---

cbSizeBarWndEvent::cbSizeBarWndEvent
cbsizebarwndeventcbsizebarwndevent
browse00390
K cbSizeBarWndEvent cbSizeBarWndEvent
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbsizebarwndevent')")
K cbSizeBarWndEvent

$#+K!cbStartBarDraggingEvent::cbStartBarDraggingEvent

cbStartBarDraggingEvent(**cbBarInfo*** *pBar*, **const wxPoint&** *pos*, **cbDockPane*** *pPane*)[K]

Constructor, taking bar information, mouse position, and pane.

---

[c]bStartBarDraggingEvent::cbStartBarDraggingEvent
[c]bstartbardraggingeventcbstartbardraggingevent
[b]rowse00392
[K] cbStartBarDraggingEvent cbStartBarDraggingEvent
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbstartbardraggingevent')")
[K] cbStartBarDraggingEvent

$#+K!**cbStartDrawInAreaEvent::cbStartDrawInAreaEvent**

cbStartDrawInAreaEvent(**const wxRect&** *area*, **wxDC**** *ppDCForArea*, **cbDockPane*** *pPane*)[K]

to the obtained buffer-context should be placed Constructor, taking rectangular area, device context pointer to a pointer, and pane.

---

[c]bStartDrawInAreaEvent::cbStartDrawInAreaEvent
[c]bstartdrawinareaeventcbstartdrawinareaevent
[b]rowse00394
[K] cbStartDrawInAreaEvent cbStartDrawInAreaEvent
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbstartdrawinareaevent')")
[K] cbStartDrawInAreaEvent

$#+K!cbUpdateMgrData::cbUpdateMgrData

cbUpdateMgrData()K

Default constructor. Is-dirty flag is set TRUE initially.

---

cbUpdateMgrData::cbUpdateMgrData
cbupdatemgrdatacbupdatemgrdata
browse00396
K cbUpdateMgrData cbUpdateMgrData
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbupdatemgrdata')")
K cbUpdateMgrData

$#+K!cbUpdateMgrData::IsDirty

bool IsDirty()K

Returns the is-dirty flag.

---

cbUpdateMgrData::IsDirty
cbupdatemgrdataisdirty
browse00397
K cbUpdateMgrData IsDirty
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbupdatemgrdata')")
K IsDirty

$#+K!cbUpdateMgrData::SetCustomData

void SetCustomData(**wxObject*** *pCustomData*)[K]

Set custom data.

---

[c]bUpdateMgrData::SetCustomData
[c]bupdatemgrdatasetcustomdata
[b]rowse00398
[K] cbUpdateMgrData SetCustomData
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbupdatemgrdata')")
[K] SetCustomData

$#+K!**cbUpdateMgrData::SetDirty**

void SetDirty(**bool** *isDirty = TRUE*)K

Set the dirty flag.

---

cbUpdateMgrData::SetDirty
cbupdatemgrdatasetdirty
browse00399
K cbUpdateMgrData SetDirty
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbupdatemgrdata')")
K SetDirty

$#+K!**cbUpdateMgrData::StoreItemState**

void StoreItemState(**const wxRect&** *boundsInParent*)[K]

Store the item state.

---

[c]bUpdateMgrData::StoreItemState
[c]bupdatemgrdatastoreitemstate
[b]rowse00400
[K] cbUpdateMgrData StoreItemState
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbupdatemgrdata')")
[K] StoreItemState

$#+K!**cbUpdatesManagerBase::cbUpdatesManagerBase**

cbUpdatesManagerBase(**wxFrameLayout*** *pPanel*)[K]

Constructor taking layout panel.

cbUpdatesManagerBase()[K]

Default constructor

---

[c]bUpdatesManagerBase::cbUpdatesManagerBase
[c]bupdatesmanagerbasecbupdatesmanagerbase
[b]rowse00402
[K] cbUpdatesManagerBase cbUpdatesManagerBase
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbupdatesmanagerbase')")
[K] cbUpdatesManagerBase
[K] cbUpdatesManagerBase

$#+K!**cbUpdatesManagerBase::~cbUpdatesManagerBase**

~cbUpdatesManagerBase()K

Destructor.

---

cbUpdatesManagerBase::~cbUpdatesManagerBase
cbupdatesmanagerbasedtor
browse00403
K cbUpdatesManagerBase ~cbUpdatesManagerBase
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbupdatesmanagerbase')")
K ~cbUpdatesManagerBase

$#+K!cbUpdatesManagerBase::OnBarWillChange

void OnBarWillChange(**cbBarInfo*** *pBar*, **cbRowInfo*** *pInRow*, **cbDockPane*** *pInPane*)K

This function receives a notification from the frame layout (in the order in which they would usually be invoked). Custom updates-managers may utilize these notifications to implement a more fine-grained updating strategy.

---

cbUpdatesManagerBase::OnBarWillChange
cbupdatesmanagerbaseonbarwillchange
browse00404
K cbUpdatesManagerBase OnBarWillChange
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbupdatesmanagerbase')")
K OnBarWillChange

$#+K!cbUpdatesManagerBase::OnFinishChanges

void OnFinishChanges()[K]

This function receives a notification from the frame layout (in the order in which they would usually be invoked). Custom updates-managers may utilize these notifications to implement a more fine-grained updating strategy.

---

[c]bUpdatesManagerBase::OnFinishChanges
[c]bupdatesmanagerbaseonfinishchanges
[b]rowse00405
[K] cbUpdatesManagerBase OnFinishChanges
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbupdatesmanagerbase')")
[K] OnFinishChanges

$#+K!cbUpdatesManagerBase::OnPaneMarginsWillChange

void OnPaneMarginsWillChange(**cbDockPane*** *pPane*)K

This function receives a notification from the frame layout (in the order in which they would usually be invoked). Custom updates-managers may utilize these notifications to implement a more fine-grained updating strategy.

---

cbUpdatesManagerBase::OnPaneMarginsWillChange
cbupdatesmanagerbaseonpanemarginswillchange
browse00406
K cbUpdatesManagerBase OnPaneMarginsWillChange
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbupdatesmanagerbase')")
K OnPaneMarginsWillChange

$#+K!cbUpdatesManagerBase::OnPaneWillChange

void OnPaneWillChange(**cbDockPane*** *pPane*)K

This function receives a notification from the frame layout (in the order in which they would usually be invoked). Custom updates-managers may utilize these notifications to implement a more fine-grained updating strategy.

---

cbUpdatesManagerBase::OnPaneWillChange
cbupdatesmanagerbaseonpanewillchange
browse00407
K cbUpdatesManagerBase OnPaneWillChange
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbupdatesmanagerbase')")
K OnPaneWillChange

$#+K!cbUpdatesManagerBase::OnRowWillChange

void OnRowWillChange(**cbRowInfo*** *pRow*, **cbDockPane*** *pInPane*)[K]

This function receives a notification from the frame layout (in the order in which they would usually be invoked). Custom updates-managers may utilize these notifications to implement a more fine-grained updating strategy.

---

[c]bUpdatesManagerBase::OnRowWillChange
[c]bupdatesmanagerbaseonrowwillchange
[b]rowse00408
[K] cbUpdatesManagerBase OnRowWillChange
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbupdatesmanagerbase')")
[K] OnRowWillChange

$#+K!cbUpdatesManagerBase::OnStartChanges

void OnStartChanges()K

This function receives a notification from the frame layout (in the order in which they would usually be invoked). Custom updates-managers may utilize these notifications to implement a more fine-grained updating strategy.

---

cbUpdatesManagerBase::OnStartChanges
cbupdatesmanagerbaseonstartchanges
browse00409
K cbUpdatesManagerBase OnStartChanges
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbupdatesmanagerbase')")
K OnStartChanges

$#+K!**cbUpdatesManagerBase::SetLayout**

void SetLayout(**wxFrameLayout*** *pLayout*)K

Sets the associated layout.

---

cbUpdatesManagerBase::SetLayout
cbupdatesmanagerbasesetlayout
browse00410
K cbUpdatesManagerBase SetLayout
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbupdatesmanagerbase')")
K SetLayout

$#+K!cbUpdatesManagerBase::UpdateNow

void UpdateNow()K

Refreshes parts of the frame layout that need an update.

---

cbUpdatesManagerBase::UpdateNow
cbupdatesmanagerbaseupdatenow
browse00411
K cbUpdatesManagerBase UpdateNow
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `cbupdatesmanagerbase')")
K UpdateNow

$#+K!**wxDynamicToolBar::wxDynamicToolBar**

wxDynamicToolBar()K

Default constructor.

wxDynamicToolBar(**wxWindow*** *parent*, **const wxWindowID** *id*, **const wxPoint&** *pos* = *wxDefaultPosition*, **const wxSize&** *size* = *wxDefaultSize*, **const long** *style* = *wxNO_BORDER*, **const int** *orientation* = *wxVERTICAL*, **const int** *RowsOrColumns* = *1*, **const wxString&** *name* = *wxToolBarNameStr*)K

Constructor: see the documentation for wxToolBar for details.

---

wxDynamicToolBar::wxDynamicToolBar
wxdynamictoolbarwxdynamictoolbar
browse00413
K wxDynamicToolBar wxDynamicToolBar
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxdynamictoolbar')")
K wxDynamicToolBar
K wxDynamicToolBar

$#+K!**wxDynamicToolBar::~wxDynamicToolBar**

~wxDynamicToolBar()K

Destructor.

---

wxDynamicToolBar::~wxDynamicToolBar
wxdynamictoolbardtor
browse00414
K wxDynamicToolBar ~wxDynamicToolBar
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxdynamictoolbar')")
K ~wxDynamicToolBar

$#+K!wxDynamicToolBar::AddSeparator

void AddSeparator(**wxWindow*** *pSepartorWnd = NULL*)K

Adds a separator. See the documentation for wxToolBar for details.

---

wxDynamicToolBar::AddSeparator
wxdynamictoolbaraddseparator
browse00415
K wxDynamicToolBar AddSeparator
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxdynamictoolbar')")
K AddSeparator

$#+K!wxDynamicToolBar::AddTool

void AddTool(**int** *toolIndex*, **wxWindow*** *pToolWindow*, **const wxSize&** *size = wxDefaultSize*)[K]

Adds a tool. See the documentation for wxToolBar for details.

void AddTool(**int** *toolIndex*, **const wxString&** *imageFileName*, **wxBitmapType** *imageFileType = wxBITMAP_TYPE_BMP*, **const wxString&** *labelText = ""*, **bool** *alignTextRight = FALSE*, **bool** *isFlat = TRUE*)[K]

Adds a tool. See the documentation for wxToolBar for details.

void AddTool(**int** *toolIndex*, **wxBitmap** *labelBmp*, **const wxString&** *labelText = ""*, **bool** *alignTextRight = FALSE*, **bool** *isFlat = TRUE*)[K]

Adds a tool. See the documentation for wxToolBar for details.

wxToolBarToolBase* AddTool(**const int** *toolIndex*, **const wxBitmap&** *bitmap*, **const wxBitmap&** *pushedBitmap = wxNullBitmap*, **const bool** *toggle = FALSE*, **const long** *xPos = -1*, **const long** *yPos = -1*, **wxObject*** *clientData = NULL*, **const wxString&** *helpString1 = ""*, **const wxString&** *helpString2 = ""*)[K]

Method from wxToolBarBase (for compatibility), only the first two arguments are valid. See the documentation for wxToolBar for details.

---

[w]xDynamicToolBar::AddTool
[w]xdynamictoolbaraddtool
[b]rowse00416
[K] wxDynamicToolBar AddTool
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxdynamictoolbar')")
[K] AddTool
[K] AddTool
[K] AddTool
[K] AddTool

$#+K!wxDynamicToolBar::Create

bool Create(**wxWindow*** *parent*, **const wxWindowID** *id*, **const wxPoint&** *pos = wxDefaultPosition*, **const wxSize&** *size = wxDefaultSize*, **const long** *style = wxNO_BORDER*, **const int** *orientation = wxVERTICAL*, **const int** *RowsOrColumns = 1*, **const wxString&** *name = wxToolBarNameStr*)K

Creation function: see the documentation for wxToolBar for details.

---

wxDynamicToolBar::Create
wxdynamictoolbarcreate
browse00417
K wxDynamicToolBar Create
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxdynamictoolbar')")
K Create

$#+K!**wxDynamicToolBar::CreateDefaultLayout**

LayoutManagerBase* CreateDefaultLayout()K

Creates the default layout (BagLayout).

---

wxDynamicToolBar::CreateDefaultLayout
wxdynamictoolbarcreatedefaultlayout
browse00418
K wxDynamicToolBar CreateDefaultLayout
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxdynamictoolbar')")
K CreateDefaultLayout

$#+K!**wxDynamicToolBar::CreateTool**

wxToolBarToolBase* CreateTool(**wxControl*** *control*)K

Creates a toolbar tool.

wxToolBarToolBase* CreateTool(**int** *id*, **const wxBitmap&** *bitmap1*, **const wxBitmap&** *bitmap2*, **bool** *toggle*, **wxObject*** *clientData*, **const wxString&** *shortHelpString*, **const wxString&** *longHelpString*)K

Creates a toolbar tool.

---

wxDynamicToolBar::CreateTool
wxdynamictoolbarcreatetool
browse00419
K wxDynamicToolBar CreateTool
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxdynamictoolbar')")
K CreateTool
K CreateTool

$#+K!**wxDynamicToolBar::DoDeleteTool**

bool DoDeleteTool(**size_t** *pos*, **wxToolBarToolBase*** *tool*)K

Deletes a tool. The tool is still in m_tools list when this function is called, and it will only be deleted from it if it succeeds.

---

wxDynamicToolBar::DoDeleteTool
wxdynamictoolbardodeletetool
browse00420
K wxDynamicToolBar DoDeleteTool
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxdynamictoolbar')")
K DoDeleteTool

$#+K!**wxDynamicToolBar::DoEnableTool**

void DoEnableTool(**wxToolBarToolBase*** *tool*, **bool** *enable*)K

Called when the tools enabled flag changes.

---

wxDynamicToolBar::DoEnableTool
wxdynamictoolbardoenabletool
browse00421
K wxDynamicToolBar DoEnableTool
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxdynamictoolbar')")
K DoEnableTool

$#+K!**wxDynamicToolBar::DoInsertTool**

bool DoInsertTool(**size_t** *pos*, **wxToolBarToolBase*** *tool*)K

Inserts a tool at the given position.

---

wxDynamicToolBar::DoInsertTool
wxdynamictoolbardoinserttool
browse00422
K wxDynamicToolBar DoInsertTool
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxdynamictoolbar')")
K DoInsertTool

$#+K!wxDynamicToolBar::DoSetToggle

void DoSetToggle(**wxToolBarToolBase*** *tool*, **bool** *toggle*)K

Called when the tools 'can be toggled' flag changes.

---

wxDynamicToolBar::DoSetToggle
wxdynamictoolbardosettoggle
browse00423
K wxDynamicToolBar DoSetToggle
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxdynamictoolbar')")
K DoSetToggle

$#+K!**wxDynamicToolBar::DoToggleTool**

void DoToggleTool(**wxToolBarToolBase*** *tool*, **bool** *toggle*)[K]

Called when the tool is toggled.

---

[w]xDynamicToolBar::DoToggleTool
[w]xdynamictoolbardotoggletool
[b]rowse00424
[K] wxDynamicToolBar DoToggleTool
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxdynamictoolbar')")
[K] DoToggleTool

$#+K!**wxDynamicToolBar::DrawSeparator**

void DrawSeparator(**wxDynToolInfo&** *info*, **wxDC&** *dc*)[K]

Draws a separator. The default implementation draws a shaded line.

---

[w]xDynamicToolBar::DrawSeparator
[w]xdynamictoolbardrawseparator
[b]rowse00425
[K] wxDynamicToolBar DrawSeparator
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxdynamictoolbar')")
[K] DrawSeparator

$#+K!**wxDynamicToolBar::EnableTool**

void EnableTool(**const int** *toolIndex*, **const bool** *enable* = *TRUE*)K

Enables or disables the given tool.

---

wxDynamicToolBar::EnableTool
wxdynamictoolbarenabletool
browse00426
K wxDynamicToolBar EnableTool
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxdynamictoolbar')")
K EnableTool

$#+KK!**wxDynamicToolBar::FindToolForPosition**

wxToolBarToolBase* FindToolForPosition(**wxCoord** *x*, **wxCoord** *y*) **const**

Finds a tool for the given position.

---

wxDynamicToolBar::FindToolForPosition
wxdynamictoolbarfindtoolforposition
browse00427
K wxDynamicToolBar FindToolForPosition
K FindToolForPosition
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxdynamictoolbar')")

$#+K!wxDynamicToolBar::GetPreferredDim

void GetPreferredDim(**const wxSize&** *givenDim*, **wxSize&** *prefDim*)K

Returns the preferred dimension, taking the given dimension and a reference to the result.

---

wxDynamicToolBar::GetPreferredDim
wxdynamictoolbargetpreferreddim
browse00428
K wxDynamicToolBar GetPreferredDim
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxdynamictoolbar')")
K GetPreferredDim

$#+K!**wxDynamicToolBar::GetToolInfo**

wxDynToolInfo* GetToolInfo(**int** *toolIndex*)[K]

Returns tool information for the given tool index.

---

[w]xDynamicToolBar::GetToolInfo
[w]xdynamictoolbargettoolinfo
[b]rowse00429
[K] wxDynamicToolBar GetToolInfo
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxdynamictoolbar')")
[K] GetToolInfo

$#+K!**wxDynamicToolBar::Layout**

bool Layout()K

Performs layout. See definitions of orientation types.

---

wxDynamicToolBar::Layout
wxdynamictoolbarlayout
browse00430
K wxDynamicToolBar Layout
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxdynamictoolbar')")
K Layout

$#+K!**wxDynamicToolBar::OnEraseBackground**

void OnEraseBackground(**wxEraseEvent&** *event*)[K]

Responds to background erase events. Currently does nothing.

---

[w]xDynamicToolBar::OnEraseBackground
[w]xdynamictoolbaronerasebackground
[b]rowse00431
[K] wxDynamicToolBar OnEraseBackground
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxdynamictoolbar')")
[K] OnEraseBackground

$#+K!wxDynamicToolBar::OnPaint

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

Responds to paint events, drawing separators.

---

wxDynamicToolBar::OnPaint
wxdynamictoolbaronpaint
browse00432
K wxDynamicToolBar OnPaint
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxdynamictoolbar')")
K OnPaint

$#+K!**wxDynamicToolBar::OnSize**

void OnSize(**wxSizeEvent&** *event*)K

Responds to size events, calling Layout.

---

wxDynamicToolBar::OnSize
wxdynamictoolbaronsize
browse00433
K wxDynamicToolBar OnSize
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxdynamictoolbar')")
K OnSize

$#+K!wxDynamicToolBar::Realize

bool Realize()[K]

Overriden from wxToolBarBase; does nothing.

---

[w]xDynamicToolBar::Realize
[w]xdynamictoolbarrealize
[b]rowse00434
[K] wxDynamicToolBar Realize
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxdynamictoolbar')")
[K] Realize

$#+K!**wxDynamicToolBar::RemveTool**

void RemveTool(**int** *toolIndex*)[K]

Removes the given tool. Misspelt in order not to clash with a similar function in the base class.

---

[w]xDynamicToolBar::RemveTool
[w]xdynamictoolbarremvetool
[b]rowse00435
[K] wxDynamicToolBar RemveTool
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxdynamictoolbar')")
[K] RemveTool

$#+K!**wxDynamicToolBar::SetLayout**

void SetLayout(**LayoutManagerBase*** *pLayout*)K

Sets the layout for this toolbar.

---

wxDynamicToolBar::SetLayout
wxdynamictoolbarsetlayout
browse00436
K wxDynamicToolBar SetLayout
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxdynamictoolbar')")
K SetLayout

$#+K!**wxDynamicToolBar::SizeToolWindows**

void SizeToolWindows()K

Internal function for sizing tool windows.

---

wxDynamicToolBar::SizeToolWindows
wxdynamictoolbarsizetoolwindows
browse00437
K wxDynamicToolBar SizeToolWindows
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxdynamictoolbar')")
K SizeToolWindows

$#+K!**wxFrameLayout::wxFrameLayout**

wxFrameLayout(**wxWindow*** *pParentFrame*, **wxWindow*** *pFrameClient = NULL*, **bool** *activateNow = TRUE*)K

Constructor, taking parent window, the (MDI) client of the parent if there is one, and flag specifying whether to activate the layout.

wxFrameLayout()K

Default constructor, used only for serialization.

---

wxFrameLayout::wxFrameLayout
wxframelayoutwxframelayout
browse00440
K wxFrameLayout wxFrameLayout
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxframelayout')")
K wxFrameLayout
K wxFrameLayout

$#+K!**wxFrameLayout::~wxFrameLayout**

~wxFrameLayout()K

Destructor. It does not destroy the bar windows.

---

wxFrameLayout::~wxFrameLayout
wxframelayoutdtor
browse00441
K wxFrameLayout ~wxFrameLayout
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxframelayout')")
K ~wxFrameLayout

$#+K!**wxFrameLayout::Activate**

void Activate()K

Activate can be called after some other layout has been deactivated, and this one must take over the current contents of the frame window. Effectively hooks itself to the frame window, re-displays all non-hidden bar windows and repaints the decorations.

---

wxFrameLayout::Activate
wxframelayoutactivate
browse00442
K wxFrameLayout Activate
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxframelayout')")
K Activate

$#+K!**wxFrameLayout::AddBar**

void AddBar(**wxWindow*** *pBarWnd*, **const cbDimInfo&** *dimInfo*, **int** *alignment = FL_ALIGN_TOP*, **int** *rowNo = 0*, **int** *columnPos = 0*, **const wxString&** *name = "bar"*, **bool** *spyEvents = FALSE*, **int** *state = wxCBAR_DOCKED_HORIZONTALLY*)K

Adds bar information to the frame layout. The appearance of the layout is not refreshed immediately; RefreshNow() can be called if necessary. Notes: the argument pBarWnd can by NULL, resulting in bar decorations to be drawn around the empty rectangle (filled with default background colour). Argument dimInfo can be reused for adding any number of bars, since it is not used directly - instead its members are copied. If the dimensions handler is present, its instance is shared (reference counted). The dimension handler should always be allocated on the heap. pBarWnd is the window to be managed. dimInfo contains dimension information. alignment is a value such as FL_ALIGN_TOP. rowNo is the vertical position or row in the pane (if in docked state). columnPos is the horizontal position within the row in pixels (if in docked state). name is a name by which the bar can be referred in layout customization dialogs. If spyEvents is TRUE, input events for the bar should be "spyed" in order to forward unhandled mouse clicks to the frame layout, for example to enable easy draggablity of toolbars just by clicking on their interior regions. For widgets like text/tree control this value should be FALSE, since there's no certain way to detect whether the event was actually handled. state is the initial state, such as wxCBAR_DOCKED_HORIZONTALLY, wxCBAR_FLOATING, wxCBAR_HIDDEN.

---

wxFrameLayout::AddBar
wxframelayoutaddbar
browse00443
K wxFrameLayout AddBar
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxframelayout')")
K AddBar

$#+K!**wxFrameLayout::AddPlugin**

void AddPlugin(**wxClassInfo*** *pPlInfo*, **int** *paneMask* = *wxALL_PANES*)K

An advanced methods for plugin configuration using their dynamic class information, for example CLASSINFO(pluginClass). First checks if the plugin of the given class is already "hooked up". If not, adds it to the top of the plugins chain.

---

wxFrameLayout::AddPlugin
wxframelayoutaddplugin
browse00444
K wxFrameLayout AddPlugin
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxframelayout')")
K AddPlugin

$#+K!wxFrameLayout::AddPluginBefore

void AddPluginBefore(**wxClassInfo*** *pNextPlInfo*, **wxClassInfo*** *pPlInfo*, **int** *paneMask = wxALL_PANES*)K

First checks if the plugin of the given class is already hooked. If so, removes it, and then inserts it into the chain before the plugin of the class given by pNextPlInfo. Note: this method is handy in some cases where the order of the plugin-chain could be important, for example when one plugin overrides some functionality of another already-hooked plugin, so that the former plugin should be hooked before the one whose functionality is being overridden.

---

wxFrameLayout::AddPluginBefore
wxframelayoutaddpluginbefore
browse00445
K wxFrameLayout AddPluginBefore
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxframelayout')")
K AddPluginBefore

$#+K!**wxFrameLayout::ApplyBarProperties**

void ApplyBarProperties(**cbBarInfo*** *pBar*)K

Reflects changes in bar information structure visually. For example, moves the bar, changes its dimension information, or changes the pane to which it is docked.

---

wxFrameLayout::ApplyBarProperties
wxframelayoutapplybarproperties
browse00446
K wxFrameLayout ApplyBarProperties
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxframelayout')")
K ApplyBarProperties

$#+K!wxFrameLayout::CanReparent

bool CanReparent()K

Returns TRUE if the platform allows reparenting. This may not return TRUE for all platforms. Reparenting allows control bars to be floated.

---

wxFrameLayout::CanReparent
wxframelayoutcanreparent
browse00447
K wxFrameLayout CanReparent
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxframelayout')")
K CanReparent

$#+K!wxFrameLayout::CaptureEventsForPane

void CaptureEventsForPane(**cbDockPane*** *toPane*)K

Called by plugins; also captures the mouse in the parent frame.

---

wxFrameLayout::CaptureEventsForPane
wxframelayoutcaptureeventsforpane
browse00448
K wxFrameLayout CaptureEventsForPane
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxframelayout')")
K CaptureEventsForPane

$#+K!**wxFrameLayout::CaptureEventsForPlugin**

void CaptureEventsForPlugin(**cbPluginBase*** *pPlugin*)K

Captures user input events for the given plugin. Input events are: mouse movement, mouse clicks, keyboard input.

---

wxFrameLayout::CaptureEventsForPlugin
wxframelayoutcaptureeventsforplugin
browse00449
K wxFrameLayout CaptureEventsForPlugin
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxframelayout')")
K CaptureEventsForPlugin

$#+K!**wxFrameLayout::CreateCursors**

void CreateCursors()K

Creates the cursors.

---

wxFrameLayout::CreateCursors
wxframelayoutcreatecursors
browse00450
K wxFrameLayout CreateCursors
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxframelayout')")
K CreateCursors

$#+K!**wxFrameLayout::CreateUpdatesManager**

cbUpdatesManagerBase* CreateUpdatesManager()K

Returns a new cbGCUpdatesMgr object.

---

wxFrameLayout::CreateUpdatesManager
wxframelayoutcreateupdatesmanager
browse00451
K wxFrameLayout CreateUpdatesManager
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxframelayout')")
K CreateUpdatesManager

$#+K!**wxFrameLayout::Deactivate**

void Deactivate()K

Deactivate unhooks itself from frame window, and hides all non-hidden windows. Note: two frame layouts should not be active at the same time in the same frame window, since it would cause messy overlapping of bar windows from both layouts.

---

wxFrameLayout::Deactivate
wxframelayoutdeactivate
browse00452
K wxFrameLayout Deactivate
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxframelayout')")
K Deactivate

$#+K!**wxFrameLayout::DestroyBarWindows**

void DestroyBarWindows()K

Destroys the bar windows.

---

wxFrameLayout::DestroyBarWindows
wxframelayoutdestroybarwindows
browse00453
K wxFrameLayout DestroyBarWindows
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxframelayout')")
K DestroyBarWindows

$#+K!**wxFrameLayout::DoSetBarState**

void DoSetBarState(**cbBarInfo*** *pBar*)K

Applies the state to the window objects.

---

wxFrameLayout::DoSetBarState
wxframelayoutdosetbarstate
browse00454
K wxFrameLayout DoSetBarState
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxframelayout')")
K DoSetBarState

$#+K!**wxFrameLayout::EnableFloating**

void EnableFloating(**bool** *enable = TRUE*)K

Enables floating behaviour. By default floating of control bars is on.

---

wxFrameLayout::EnableFloating
wxframelayoutenablefloating
browse00455
K wxFrameLayout EnableFloating
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxframelayout')")
K EnableFloating

$#+K!**wxFrameLayout::FindBarByName**

cbBarInfo* FindBarByName(**const wxString&** *name*)[K]

Finds the bar in the framelayout, by name.

---

[w]xFrameLayout::FindBarByName
[w]xframelayoutfindbarbyname
[b]rowse00456
[K] wxFrameLayout FindBarByName
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxframelayout')")
[K] FindBarByName

$#+K!**wxFrameLayout::FindBarByWindow**

cbBarInfo* FindBarByWindow(**const wxWindow*** *pWnd*)K

Finds the bar in the framelayout, by window.

---

wxFrameLayout::FindBarByWindow
wxframelayoutfindbarbywindow
browse00457
K wxFrameLayout FindBarByWindow
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxframelayout')")
K FindBarByWindow

$#+K!**wxFrameLayout::FindPlugin**

cbPluginBase* FindPlugin(**wxClassInfo*** *pPlInfo*)K

Finds a plugin with the given class, or returns NULL if a plugin of the given class is not hooked.

---

wxFrameLayout::FindPlugin
wxframelayoutfindplugin
browse00458
K wxFrameLayout FindPlugin
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxframelayout')")
K FindPlugin

$#+K!wxFrameLayout::FirePluginEvent

void FirePluginEvent(**cbPluginEvent&** *event*)[K]

This function should be used instead of passing the event to the ProcessEvent method of the top-level plugin directly. This method checks if events are currently captured and ensures that plugin-event is routed correctly.

---

[w]xFrameLayout::FirePluginEvent
[w]xframelayoutfirepluginevent
[b]rowse00459
[K] wxFrameLayout FirePluginEvent
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxframelayout')")
[K] FirePluginEvent

$#+K!**wxFrameLayout::ForwardMouseEvent**

void ForwardMouseEvent(**wxMouseEvent&** *event*, **cbDockPane*** *pToPane*, **int** *eventType*)[K]

Delegated from "bar-spy".

---

[w]xFrameLayout::ForwardMouseEvent
[w]xframelayoutforwardmouseevent
[b]rowse00460
[K] wxFrameLayout ForwardMouseEvent
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxframelayout')")
[K] ForwardMouseEvent

$#+K!**wxFrameLayout::GetBarPane**

cbDockPane* GetBarPane(**cbBarInfo*** *pBar*)K

Returns the pane to which the given bar belongs.

---

wxFrameLayout::GetBarPane
wxframelayoutgetbarpane
browse00461
K wxFrameLayout GetBarPane
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxframelayout')")
K GetBarPane

$#+K!**wxFrameLayout::GetBars**

BarArrayT& GetBars()K

Gets an array of bars.

---

wxFrameLayout::GetBars
wxframelayoutgetbars
browse00462
K wxFrameLayout GetBars
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxframelayout')")
K GetBars

$#+K!**wxFrameLayout::GetClientHeight**

int GetClientHeight()K

Returns the client height.

---

wxFrameLayout::GetClientHeight
wxframelayoutgetclientheight
browse00463
K wxFrameLayout GetClientHeight
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxframelayout')")
K GetClientHeight

$#+K!**wxFrameLayout::GetClientRect**

wxRect& GetClientRect()K

Returns the client's rectangle.

---

wxFrameLayout::GetClientRect
wxframelayoutgetclientrect
browse00464
K wxFrameLayout GetClientRect
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxframelayout')")
K GetClientRect

$#+K!**wxFrameLayout::GetClientWidth**

int GetClientWidth()K

Returns the client width.

---

wxFrameLayout::GetClientWidth
wxframelayoutgetclientwidth
browse00465
K wxFrameLayout GetClientWidth
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxframelayout')")
K GetClientWidth

$#+K!wxFrameLayout::GetFrameClient

wxWindow* GetFrameClient()K

Returns the frame client, or NULL if not present.

---

wxFrameLayout::GetFrameClient
wxframelayoutgetframeclient
browse00466
K wxFrameLayout GetFrameClient
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxframelayout')")
K GetFrameClient

$#+K!**wxFrameLayout::GetPane**

cbDockPane* GetPane(**int** *alignment*)K

Returns a pane for the given alignment. See pane alignment types.

---

wxFrameLayout::GetPane
wxframelayoutgetpane
browse00467
K wxFrameLayout GetPane
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxframelayout')")
K GetPane

$#+K!wxFrameLayout::GetPaneProperties

void GetPaneProperties(**cbCommonPaneProperties&** *props*, **int** *alignment* = *FL_ALIGN_TOP*)K

Gets the pane properties for the given alignment.

---

wxFrameLayout::GetPaneProperties
wxframelayoutgetpaneproperties
browse00468
K wxFrameLayout GetPaneProperties
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxframelayout')")
K GetPaneProperties

$#+K!wxFrameLayout::GetPanesArray

cbDockPane GetPanesArray**()[K]

Returns an array of panes. Used by update managers.

---

[w]xFrameLayout::GetPanesArray
[w]xframelayoutgetpanesarray
[b]rowse00469
[K] wxFrameLayout GetPanesArray
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxframelayout')")
[K] GetPanesArray

$#+K!**wxFrameLayout::GetParentFrame**

wxWindow& GetParentFrame()K

Returns the parent frame.

---

wxFrameLayout::GetParentFrame
wxframelayoutgetparentframe
browse00470
K wxFrameLayout GetParentFrame
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxframelayout')")
K GetParentFrame

$#+K!wxFrameLayout::GetPrevClientRect

wxRect& GetPrevClientRect()K

Returns the previous client window rectangle.

---

wxFrameLayout::GetPrevClientRect
wxframelayoutgetprevclientrect
browse00471
K wxFrameLayout GetPrevClientRect
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxframelayout')")
K GetPrevClientRect

$#+K!**wxFrameLayout::GetTopPlugin**

cbPluginBase& GetTopPlugin()K

Returns the current top-level plugin (the one that receives events first, except if input events are currently captured by some other plugin).

---

wxFrameLayout::GetTopPlugin
wxframelayoutgettopplugin
browse00472
K wxFrameLayout GetTopPlugin
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxframelayout')")
K GetTopPlugin

$#+K!**wxFrameLayout::GetUpdatesManager**

cbUpdatesManagerBase& GetUpdatesManager()K

Returns a reference to the updates manager. Note: in future, the updates manager will become a normal plugin.

---

wxFrameLayout::GetUpdatesManager
wxframelayoutgetupdatesmanager
browse00473
K wxFrameLayout GetUpdatesManager
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxframelayout')")
K GetUpdatesManager

$#+K!wxFrameLayout::HasTopPlugin

bool HasTopPlugin()K

Returns true if there is a top plugin.

---

wxFrameLayout::HasTopPlugin
wxframelayouthastopplugin
browse00474
K wxFrameLayout HasTopPlugin
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxframelayout')")
K HasTopPlugin

$#+K!**wxFrameLayout::HideBarWindows**

void HideBarWindows()K

Hides the bar windows, and also the client window if present.

---

wxFrameLayout::HideBarWindows
wxframelayouthidebarwindows
browse00475
K wxFrameLayout HideBarWindows
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxframelayout')")
K HideBarWindows

$#+K!**wxFrameLayout::HitTestPane**

bool HitTestPane(**cbDockPane*** *pPane*, **int** *x*, **int** *y*)[K]

Returns TRUE if the position is within the given pane.

---

[w]xFrameLayout::HitTestPane
[w]xframelayouthittestpane
[b]rowse00476
[K] wxFrameLayout HitTestPane
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxframelayout')")
[K] HitTestPane

$#+K!wxFrameLayout::HitTestPanes

cbDockPane* HitTestPanes(**const wxRect&** *rect*, **cbDockPane*** *pCurPane*)K

Returns the pane for which the rectangle hit test succeeds, giving preference to the given pane if supplied.

---

wxFrameLayout::HitTestPanes
wxframelayouthittestpanes
browse00477
K wxFrameLayout HitTestPanes
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxframelayout')")
K HitTestPanes

$#+K!**wxFrameLayout::HookUpToFrame**

void HookUpToFrame()K

Hooks the layout up to the frame (pushes the layout onto the frame's event handler stack).

---

wxFrameLayout::HookUpToFrame
wxframelayouthookuptoframe
browse00478
K wxFrameLayout HookUpToFrame
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxframelayout')")
K HookUpToFrame

$#+K!**wxFrameLayout::InverseVisibility**

void InverseVisibility(**cbBarInfo*** *pBar*)K

Toggles the bar between visible and hidden.

---

wxFrameLayout::InverseVisibility
wxframelayoutinversevisibility
browse00479
K wxFrameLayout InverseVisibility
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxframelayout')")
K InverseVisibility

$#+K!**wxFrameLayout::LocateBar**

bool LocateBar(**cbBarInfo*** *pBarInfo*, **cbRowInfo**** *ppRow*, **cbDockPane**** *ppPane*)K

The purpose of this function is unknown.

---

wxFrameLayout::LocateBar
wxframelayoutlocatebar
browse00480
K wxFrameLayout LocateBar
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxframelayout')")
K LocateBar

$#+K!**wxFrameLayout::OnActivate**

void OnActivate(**wxActivateEvent&** *event*)K

Handles activation events. Currently does nothing.

---

wxFrameLayout::OnActivate
wxframelayoutonactivate
browse00481
K wxFrameLayout OnActivate
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxframelayout')")
K OnActivate

$#+K!**wxFrameLayout::OnEraseBackground**

void OnEraseBackground(**wxEraseEvent&** *event*)K

Handles background erase events. Currently does nothing.

---

wxFrameLayout::OnEraseBackground
wxframelayoutonerasebackground
browse00482
K wxFrameLayout OnEraseBackground
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxframelayout')")
K OnEraseBackground

$#+K!wxFrameLayout::OnIdle

void OnIdle(**wxIdleEvent&** *event*)K

Handles idle events.

---

wxFrameLayout::OnIdle
wxframelayoutonidle
browse00483
K wxFrameLayout OnIdle
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxframelayout')")
K OnIdle

$#+K!wxFrameLayout::OnKillFocus

void OnKillFocus(**wxFocusEvent&** *event*)K

Handles focus kill events. Currently does nothing.

---

wxFrameLayout::OnKillFocus
wxframelayoutonkillfocus
browse00484
K wxFrameLayout OnKillFocus
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxframelayout')")
K OnKillFocus

$#+K!wxFrameLayout::OnLButtonDown

void OnLButtonDown(**wxMouseEvent&** *event*)K

Event handler for a left down button event.

---

wxFrameLayout::OnLButtonDown
wxframelayoutonlbuttondown
browse00485
K wxFrameLayout OnLButtonDown
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxframelayout')")
K OnLButtonDown

$#+K!**wxFrameLayout::OnLButtonUp**

void OnLButtonUp(**wxMouseEvent&** *event*)K

Event handler for a left button up event.

---

wxFrameLayout::OnLButtonUp
wxframelayoutonlbuttonup
browse00486
K wxFrameLayout OnLButtonUp
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxframelayout')")
K OnLButtonUp

$#+K!**wxFrameLayout::OnLDblClick**

void OnLDblClick(**wxMouseEvent&** *event*)K

Event handler for a left doubleclick button event.

---

wxFrameLayout::OnLDblClick
wxframelayoutonldblclick
browse00487
K wxFrameLayout OnLDblClick
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxframelayout')")
K OnLDblClick

$#+K!**wxFrameLayout::OnMouseMove**

void OnMouseMove(**wxMouseEvent&** *event*)K

Event handler for a mouse move event.

---

wxFrameLayout::OnMouseMove
wxframelayoutonmousemove
browse00488
K wxFrameLayout OnMouseMove
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxframelayout')")
K OnMouseMove

$#+K! wxFrameLayout::OnPaint

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

Handles paint events, calling PaintPane for each pane.

---

wxFrameLayout::OnPaint
wxframelayoutonpaint
browse00489
K wxFrameLayout OnPaint
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxframelayout')")
K OnPaint

$#+K!wxFrameLayout::OnRButtonDown

void OnRButtonDown(**wxMouseEvent&** *event*)K

Event handler for a right button down event.

---

wxFrameLayout::OnRButtonDown
wxframelayoutonrbuttondown
browse00490
K wxFrameLayout OnRButtonDown
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxframelayout')")
K OnRButtonDown

$#+K!**wxFrameLayout::OnRButtonUp**

void OnRButtonUp(**wxMouseEvent&** *event*)K

Event handler for a right button up event.

---

wxFrameLayout::OnRButtonUp
wxframelayoutonrbuttonup
browse00491
K wxFrameLayout OnRButtonUp
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxframelayout')")
K OnRButtonUp

$#+K!**wxFrameLayout::OnSetFocus**

void OnSetFocus(**wxFocusEvent&** *event*)K

Handles focus set events. Currently does nothing.

---

wxFrameLayout::OnSetFocus
wxframelayoutonsetfocus
browse00492
K wxFrameLayout OnSetFocus
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxframelayout')")
K OnSetFocus

$#+K!**wxFrameLayout::OnSize**

void OnSize(**wxSizeEvent&** *event*)K

Event handler for a size event.

---

wxFrameLayout::OnSize
wxframelayoutonsize
browse00493
K wxFrameLayout OnSize
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxframelayout')")
K OnSize

$#+K!**wxFrameLayout::PopAllPlugins**

void PopAllPlugins()K

Pop all plugins.

---

wxFrameLayout::PopAllPlugins
wxframelayoutpopallplugins
browse00494
K wxFrameLayout PopAllPlugins
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxframelayout')")
K PopAllPlugins

$#+K!**wxFrameLayout::PopPlugin**

void PopPlugin()K

Similar to wxWindow's "push/pop-event-handler" methods, execept that the plugin is deleted upon "popping".

---

wxFrameLayout::PopPlugin
wxframelayoutpopplugin
browse00495
K wxFrameLayout PopPlugin
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxframelayout')")
K PopPlugin

$#+K!**wxFrameLayout::PositionClientWindow**

void PositionClientWindow()[K]

Called to apply the calculated layout to window objects.

---

[w]xFrameLayout::PositionClientWindow
[w]xframelayoutpositionclientwindow
[b]rowse00496
[K] wxFrameLayout PositionClientWindow
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxframelayout')")
[K] PositionClientWindow

$#+K!**wxFrameLayout::PositionPanes**

void PositionPanes()[K]

Called to apply the calculated layout to window objects.

---

[w]xFrameLayout::PositionPanes
[w]xframelayoutpositionpanes
[b]rowse00497
[K] wxFrameLayout PositionPanes
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxframelayout')")
[K] PositionPanes

$#+K!wxFrameLayout::PushDefaultPlugins

void PushDefaultPlugins()K

Adds the default plugins. These are cbPaneDrawPlugin, cbRowLayoutPlugin, cbBarDragPlugin, cbAntiflickerPlugin, cbSimpleCustomizePlugin. This method is automatically invoked if no plugins were found upon firing of the first plugin-event, i.e. when wxFrameLayout configures itself.

---

wxFrameLayout::PushDefaultPlugins
wxframelayoutpushdefaultplugins
browse00498
K wxFrameLayout PushDefaultPlugins
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxframelayout')")
K PushDefaultPlugins

$#+K!wxFrameLayout::PushPlugin

void PushPlugin(**cbPluginBase*** *pPugin*)K

Similar to wxWindow's "push/pop-event-handler" methods, execept that the plugin is deleted upon "popping".

---

wxFrameLayout::PushPlugin
wxframelayoutpushplugin
browse00499
K wxFrameLayout PushPlugin
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxframelayout')")
K PushPlugin

$#+K!**wxFrameLayout::RecalcLayout**

void RecalcLayout(**bool** *repositionBarsNow = FALSE*)K

Recalculates the layout of panes, and all bars/rows in each pane.

---

wxFrameLayout::RecalcLayout
wxframelayoutrecalclayout
browse00500
K wxFrameLayout RecalcLayout
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxframelayout')")
K RecalcLayout

$#+K!**wxFrameLayout::RedockBar**

bool RedockBar(**cbBarInfo*** *pBar*, **const wxRect&** *shapeInParent*, **cbDockPane*** *pToPane = NULL*, **bool** *updateNow = TRUE*)K

ReddockBar can be used for repositioning existing bars. The given bar is first removed from the pane it currently belongs to, and inserted into the pane, which "matches" the given rectangular area. If pToPane is not NULL, the bar is docked to this given pane. To dock a bar which is floating, use the wxFrameLayout::DockBar method.

---

wxFrameLayout::RedockBar
wxframelayoutredockbar
browse00501
K wxFrameLayout RedockBar
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxframelayout')")
K RedockBar

$#+K!**wxFrameLayout::RefreshNow**

void RefreshNow(**bool** *recalcLayout = TRUE*)K

Recalculates layout and performs on-screen update of all panes.

---

wxFrameLayout::RefreshNow
wxframelayoutrefreshnow
browse00502
K wxFrameLayout RefreshNow
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxframelayout')")
K RefreshNow

$#+K!wxFrameLayout::ReleaseEventsFromPane

void ReleaseEventsFromPane(**cbDockPane*** *fromPane*)[K]

Called by plugins; also releases mouse in the parent frame.

---

[w]xFrameLayout::ReleaseEventsFromPane
[w]xframelayoutreleaseeventsfrompane
[b]rowse00503
[K] wxFrameLayout ReleaseEventsFromPane
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxframelayout')")
[K] ReleaseEventsFromPane

$#+K!**wxFrameLayout::ReleaseEventsFromPlugin**

void ReleaseEventsFromPlugin(**cbPluginBase*** *pPlugin*)K

Releases user input events for the given plugin. Input events are: mouse movement, mouse clicks, keyboard input

---

wxFrameLayout::ReleaseEventsFromPlugin
wxframelayoutreleaseeventsfromplugin
browse00504
K wxFrameLayout ReleaseEventsFromPlugin
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxframelayout')")
K ReleaseEventsFromPlugin

$#+K!**wxFrameLayout::RemoveBar**

void RemoveBar(**cbBarInfo*** *pBar*)K

Removes the bar from the layout permanently, and hides its corresponding window if present.

---

wxFrameLayout::RemoveBar
wxframelayoutremovebar
browse00505
K wxFrameLayout RemoveBar
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxframelayout')")
K RemoveBar

$#+K!**wxFrameLayout::RemovePlugin**

void RemovePlugin(**wxClassInfo*** *pPlInfo*)K

Checks if the plugin of the given class is hooked, and removes it if found.

---

wxFrameLayout::RemovePlugin
wxframelayoutremoveplugin
browse00506
K wxFrameLayout RemovePlugin
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxframelayout')")
K RemovePlugin

$#+K!**wxFrameLayout::ReparentWindow**

void ReparentWindow(**wxWindow*** *pChild*, **wxWindow*** *pNewParent*)K

Reparents pChild to have parent pNewParent.

---

wxFrameLayout::ReparentWindow
wxframelayoutreparentwindow
browse00507
K wxFrameLayout ReparentWindow
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxframelayout')")
K ReparentWindow

$#+K!**wxFrameLayout::RepositionFloatedBar**

void RepositionFloatedBar(**cbBarInfo*** *pBar*)K

Applies the calculated layout to a floating bar.

---

wxFrameLayout::RepositionFloatedBar
wxframelayoutrepositionfloatedbar
browse00508
K wxFrameLayout RepositionFloatedBar
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxframelayout')")
K RepositionFloatedBar

$#+K!**wxFrameLayout::RouteMouseEvent**

void RouteMouseEvent(**wxMouseEvent&** *event*, **int** *pluginEvtType*)K

Routes the mouse event to the appropriate pane.

---

wxFrameLayout::RouteMouseEvent
wxframelayoutroutemouseevent
browse00509
K wxFrameLayout RouteMouseEvent
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxframelayout')")
K RouteMouseEvent

$#+K!**wxFrameLayout::SetBarState**

void SetBarState(**cbBarInfo*** *pBar*, **int** *newStatem*, **bool** *updateNow*)K

Changes the bar's docking state (see possible control bar states).

---

wxFrameLayout::SetBarState
wxframelayoutsetbarstate
browse00510
K wxFrameLayout SetBarState
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxframelayout')")
K SetBarState

$#+K!**wxFrameLayout::SetFrameClient**

void SetFrameClient(**wxWindow*** *pFrameClient*)K

Passes the client window (e.g. MDI client window) to be controlled by frame layout, the size and position of which should be adjusted to be surrounded by controlbar panes, whenever the frame is resized or the dimensions of control panes change.

---

wxFrameLayout::SetFrameClient
wxframelayoutsetframeclient
browse00511
K wxFrameLayout SetFrameClient
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxframelayout')")
K SetFrameClient

$#+K!wxFrameLayout::SetMargins

void SetMargins(**int** *top*, **int** *bottom*, **int** *left*, **int** *right*, **int** *paneMask = wxALL_PANES*)K

Sets the margins for the given panes. The margins should go into cbCommonPaneProperties in the future. Note: this method should be called before any custom plugins are attached.

---

wxFrameLayout::SetMargins
wxframelayoutsetmargins
browse00512
K wxFrameLayout SetMargins
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxframelayout')")
K SetMargins

$#+K!**wxFrameLayout::SetPaneBackground**

void SetPaneBackground(**const wxColour&** *colour*)K

Sets the pane background colour.

---

wxFrameLayout::SetPaneBackground
wxframelayoutsetpanebackground
browse00513
K wxFrameLayout SetPaneBackground
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxframelayout')")
K SetPaneBackground

$#+K! wxFrameLayout::SetPaneProperties

void SetPaneProperties(**const cbCommonPaneProperties&** *props*, **int** *paneMask* = *wxALL_PANES*)K

Sets the pane properties for the given alignment. Note: changing properties of panes does not result immediate on-screen update.

---

wxFrameLayout::SetPaneProperties
wxframelayoutsetpaneproperties
browse00514
K wxFrameLayout SetPaneProperties
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxframelayout')")
K SetPaneProperties

$#+K!wxFrameLayout::SetTopPlugin

void SetTopPlugin(**cbPluginBase*** *pPlugin*)K

Hooking custom plugins to frame layout. Note: when hooking one plugin on top of the other, use SetNextHandler or similar methods of wxEvtHandler class to compose the chain of plugins, than pass the left-most handler in this chain to the above methods (assuming that events are delegated from left-most towards right-most handler). This secenario is very inconvenient and "low-level", so use the Add/Push/PopPlugin methods instead.

---

wxFrameLayout::SetTopPlugin
wxframelayoutsettopplugin
browse00515
K wxFrameLayout SetTopPlugin
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxframelayout')")
K SetTopPlugin

$#+K!**wxFrameLayout::SetUpdatesManager**

void SetUpdatesManager(**cbUpdatesManagerBase*** *pUMgr*)[K]

Destroys the previous manager if any, and sets the new one.

---

[w]xFrameLayout::SetUpdatesManager
[w]xframelayoutsetupdatesmanager
[b]rowse00516
[K] wxFrameLayout SetUpdatesManager
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxframelayout')")
[K] SetUpdatesManager

$#+K!**wxFrameLayout::ShowFloatedWindows**

void ShowFloatedWindows(**bool** *show*)K

Shows all floated windows.

---

wxFrameLayout::ShowFloatedWindows
wxframelayoutshowfloatedwindows
browse00517
K wxFrameLayout ShowFloatedWindows
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxframelayout')")
K ShowFloatedWindows

$#+K!**wxFrameLayout::UnhookFromFrame**

void UnhookFromFrame()K

Unhooks the layout from the frame.

---

wxFrameLayout::UnhookFromFrame
wxframelayoutunhookfromframe
browse00518
K wxFrameLayout UnhookFromFrame
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxframelayout')")
K UnhookFromFrame

$#+K!**wxFrameManager::wxFrameManager**

wxFrameManager()K

---

wxFrameManager::wxFrameManager
wxframemanagerwxframemanager
browse00520
K wxFrameManager wxFrameManager
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxframemanager')")
K wxFrameManager

$#+K!**wxFrameManager::~wxFrameManager**

~wxFrameManager()K

---

wxFrameManager::~wxFrameManager
wxframemanagerdtor
browse00521
K wxFrameManager ~wxFrameManager
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxframemanager')")
K ~wxFrameManager

$#+K!**wxFrameManager::ActivateView**

void ActivateView(**wxFrameView*** *pFrmView*)K

void ActivateView(**int** *viewNo*)K

---

wxFrameManager::ActivateView
wxframemanageractivateview
browse00522
K wxFrameManager ActivateView
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxframemanager')")
K ActivateView
K ActivateView

$#+K!**wxFrameManager::AddView**

void AddView(**wxFrameView*** *pFrmView*)K

---

wxFrameManager::AddView
wxframemanageraddview
browse00523
K wxFrameManager AddView
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxframemanager')")
K AddView

$#+K!**wxFrameManager::DeactivateCurrentView**

void DeactivateCurrentView()K

---

wxFrameManager::DeactivateCurrentView
wxframemanagerdeactivatecurrentview
browse00524
K wxFrameManager DeactivateCurrentView
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxframemanager')")
K DeactivateCurrentView

$#+K!**wxFrameManager::DestroyViews**

void DestroyViews()K

---

wxFrameManager::DestroyViews
wxframemanagerdestroyviews
browse00525
K wxFrameManager DestroyViews
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxframemanager')")
K DestroyViews

$#+K!**wxFrameManager::DoSerialize**

void DoSerialize(**wxObjectStorage&** *store*)K

---

wxFrameManager::DoSerialize
wxframemanagerdoserialize
browse00526
K wxFrameManager DoSerialize
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxframemanager')")
K DoSerialize

$#+K!**wxFrameManager::EnableMenusForView**

void EnableMenusForView(**wxFrameView*** *pView*, **bool** *enable*)K

---

wxFrameManager::EnableMenusForView
wxframemanagerenablemenusforview
browse00527
K wxFrameManager EnableMenusForView
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxframemanager')")
K EnableMenusForView

$#+K!**wxFrameManager::GetActiveView**

wxFrameView* GetActiveView()K

---

wxFrameManager::GetActiveView
wxframemanagergetactiveview
browse00528
K wxFrameManager GetActiveView
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxframemanager')")
K GetActiveView

$#+K!**wxFrameManager::GetActiveViewNo**

int GetActiveViewNo()K

---

wxFrameManager::GetActiveViewNo
wxframemanagergetactiveviewno
browse00529
K wxFrameManager GetActiveViewNo
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxframemanager')")
K GetActiveViewNo

$#+K!**wxFrameManager::GetActiveViewNode**

wxNode* GetActiveViewNode()K

---

wxFrameManager::GetActiveViewNode
wxframemanagergetactiveviewnode
browse00530
K wxFrameManager GetActiveViewNode
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxframemanager')")
K GetActiveViewNode

$#+K!**wxFrameManager::GetClientWindow**

wxWindow* GetClientWindow()K

---

wxFrameManager::GetClientWindow
wxframemanagergetclientwindow
browse00531
K wxFrameManager GetClientWindow
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxframemanager')")
K GetClientWindow

$#+K!**wxFrameManager::GetObjectStore**

wxObjectStorage& GetObjectStore()K

---

wxFrameManager::GetObjectStore
wxframemanagergetobjectstore
browse00532
K wxFrameManager GetObjectStore
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxframemanager')")
K GetObjectStore

$#+K!**wxFrameManager::GetParentFrame**

wxFrame* GetParentFrame()K

synonyms

---

wxFrameManager::GetParentFrame
wxframemanagergetparentframe
browse00533
K wxFrameManager GetParentFrame
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxframemanager')")
K GetParentFrame

$#+K!**wxFrameManager::GetParentWindow**

wxWindow* GetParentWindow()K

---

wxFrameManager::GetParentWindow
wxframemanagergetparentwindow
browse00534
K wxFrameManager GetParentWindow
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxframemanager')")
K GetParentWindow

$#+K!wxFrameManager::GetView

wxFrameView* GetView(**int** *viewNo*)K

---

wxFrameManager::GetView
wxframemanagergetview
browse00535
K wxFrameManager GetView
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxframemanager')")
K GetView

$#+K!**wxFrameManager::GetViewNo**

int GetViewNo(**wxFrameView*** *pView*)K

---

wxFrameManager::GetViewNo
wxframemanagergetviewno
browse00536
K wxFrameManager GetViewNo
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxframemanager')")
K GetViewNo

$#+K!**wxFrameManager::Init**

void Init(**wxWindow*** *pMainFrame*, **const wxString&** *settingsFile = ""*)K

if file name is empty, views are are not saved/loaded

---

wxFrameManager::Init
wxframemanagerinit
browse00537
K wxFrameManager Init
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxframemanager')")
K Init

$#+K!**wxFrameManager::ReloadViews**

bool ReloadViews()K

---

wxFrameManager::ReloadViews
wxframemanagerreloadviews
browse00538
K wxFrameManager ReloadViews
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxframemanager')")
K ReloadViews

$#+K!**wxFrameManager::RemoveView**

void RemoveView(**wxFrameView*** *pFrmView*)K

---

wxFrameManager::RemoveView
wxframemanagerremoveview
browse00539
K wxFrameManager RemoveView
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxframemanager')")
K RemoveView

$#+K!wxFrameManager::SaveViewsNow

void SaveViewsNow()K

---

wxFrameManager::SaveViewsNow
wxframemanagersaveviewsnow
browse00540
K wxFrameManager SaveViewsNow
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxframemanager')")
K SaveViewsNow

$#+K!wxFrameManager::SetClinetWindow

void SetClinetWindow(**wxWindow*** *pFrameClient*)K

---

wxFrameManager::SetClinetWindow
wxframemanagersetclinetwindow
browse00541
K wxFrameManager SetClinetWindow
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxframemanager')")
K SetClinetWindow

$#+K!**wxFrameManager::SyncAllMenus**

void SyncAllMenus()K

---

wxFrameManager::SyncAllMenus
wxframemanagersyncallmenus
browse00542
K wxFrameManager SyncAllMenus
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxframemanager')")
K SyncAllMenus

$#+K!**wxFrameManager::ViewsAreLoaded**

bool ViewsAreLoaded()K

---

wxFrameManager::ViewsAreLoaded
wxframemanagerviewsareloaded
browse00543
K wxFrameManager ViewsAreLoaded
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxframemanager')")
K ViewsAreLoaded

$#+K!**GarbageCollector::GarbageCollector**

GarbageCollector()[K]

Default constructor.

[G]arbageCollector::GarbageCollector
[g]arbagecollectorgarbagecollector
[b]rowse00545
[K] GarbageCollector GarbageCollector
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `garbagecollector')")
[K] GarbageCollector

$#+K!**GarbageCollector::~GarbageCollector**

~GarbageCollector()K

Destructor.

GarbageCollector::~GarbageCollector
garbagecollectordtor
browse00546
K GarbageCollector ~GarbageCollector
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `garbagecollector')")
K ~GarbageCollector

$#+K!**GarbageCollector::AddDependency**

void AddDependency(**void*** *pObj*, **void*** *pDependsOnObj*)K

Prepare data for garbage collection.

GarbageCollector::AddDependency
garbagecollectoradddependency
browse00547
K GarbageCollector AddDependency
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `garbagecollector')")
K AddDependency

$#+K!**GarbageCollector::AddObject**

void AddObject(**void*** *pObj*, **int** *refCnt = 1*)K

Prepare data for garbage collection.

GarbageCollector::AddObject
garbagecollectoraddobject
browse00548
K GarbageCollector AddObject
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `garbagecollector')")
K AddObject

$#+K!GarbageCollector::ArrangeCollection

void ArrangeCollection()K

Executes garbage collection algorithm.

G arbageCollector::ArrangeCollection
g arbagecollectorarrangecollection
b rowse00549
K GarbageCollector ArrangeCollection
E nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `garbagecollector')")
K ArrangeCollection

$#+K!GarbageCollector::DestroyItemList

void DestroyItemList(wxList& *lst*)[K]

Destroys a list of items.

[G]arbageCollector::DestroyItemList
[g]arbagecollectordestroyitemlist
[b]rowse00550
[K] GarbageCollector DestroyItemList
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `garbagecollector')")
[K] DestroyItemList

$#+K!GarbageCollector::FindItemNode

wxNode* FindItemNode(**void*** *pForObj*)K

Internal method for finding a node.

GarbageCollector::FindItemNode
garbagecollectorfinditemnode
browse00551
K GarbageCollector FindItemNode
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `garbagecollector')")
K FindItemNode

$#+K!GarbageCollector::FindReferenceFreeItemNode

wxNode* FindReferenceFreeItemNode()K

Internal method for findind and freeing a node.

GarbageCollector::FindReferenceFreeItemNode
garbagecollectorfindreferencefreeitemnode
browse00552
K GarbageCollector FindReferenceFreeItemNode
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `garbagecollector')")
K FindReferenceFreeItemNode

$#+K!**GarbageCollector::GetCycledObjects**

wxList& GetCycledObjects()K

Get cycled objects.

GarbageCollector::GetCycledObjects
garbagecollectorgetcycledobjects
browse00553
K GarbageCollector GetCycledObjects
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `garbagecollector')")
K GetCycledObjects

$#+K!GarbageCollector::GetRegularObjects

wxList& GetRegularObjects()K

Accesses the results of the algorithm.

GarbageCollector::GetRegularObjects
garbagecollectorgetregularobjects
browse00554
K GarbageCollector GetRegularObjects
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `garbagecollector')")
K GetRegularObjects

$#+K!GarbageCollector::RemoveReferencesToNode

void RemoveReferencesToNode(**wxNode*** *pItemNode*)K

Remove references to this node.

GarbageCollector::RemoveReferencesToNode
garbagecollectorremovereferencestonode
browse00555
K GarbageCollector RemoveReferencesToNode
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `garbagecollector')")
K RemoveReferencesToNode

$#+K!GarbageCollector::Reset

void Reset()K

Removes all data from the garbage collector.

GarbageCollector::Reset
garbagecollectorreset
browse00556
K GarbageCollector Reset
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `garbagecollector')")
K Reset

$#+K!GarbageCollector::ResolveReferences

void ResolveReferences()K

Internal method for resolving references.

GarbageCollector::ResolveReferences
garbagecollectorresolvereferences
browse00557
K GarbageCollector ResolveReferences
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `garbagecollector')")
K ResolveReferences

$#+K!**LayoutManagerBase::~LayoutManagerBase**

~LayoutManagerBase()K

Destructor.

Layoutmanagerbase::~LayoutManagerBase
layoutmanagerbasedtor
browse00559
K LayoutManagerBase ~LayoutManagerBase
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `layoutmanagerbase')")
K ~LayoutManagerBase

$#+K!**LayoutManagerBase::Layout**

void Layout(**const wxSize&** *parentDim*, **wxSize&** *resultingDim*, **wxLayoutItemArrayT&** *items*, **int** *horizGap*, **int** *vertGap*)K

Constructor.

LayoutManagerBase::Layout
layoutmanagerbaselayout
browse00560
K LayoutManagerBase Layout
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `layoutmanagerbase')")
K Layout

$#+K!wxNewBitmapButton::wxNewBitmapButton

wxNewBitmapButton(**const wxBitmap&** *labelBitmap = wxNullBitmap*, **const wxString&** *labelText = ""*, **int** *alignText = NB_ALIGN_TEXT_BOTTOM*, **bool** *isFlat = TRUE*, **int** *firedEventType = wxEVT_COMMAND_MENU_SELECTED*, **int** *marginX = 2*, **int** *marginY = 2*, **int** *textToLabelGap = 2*, **bool** *isSticky = FALSE*)[K]

Constructor.

wxNewBitmapButton(**const wxString&** *bitmapFileName*, **const wxBitmapType** *bitmapFileType = wxBITMAP_TYPE_BMP*, **const wxString&** *labelText = ""*, **int** *alignText = NB_ALIGN_TEXT_BOTTOM*, **bool** *isFlat = TRUE*, **int** *firedEventType = wxEVT_COMMAND_MENU_SELECTED*, **int** *marginX = 2*, **int** *marginY = 2*, **int** *textToLabelGap = 2*, **bool** *isSticky = FALSE*)[K]

Use this constructor if buttons have to be persistant

---

[w]xNewBitmapButton::wxNewBitmapButton
[w]xnewbitmapbuttonwxnewbitmapbutton
[b]rowse00562
[K] wxNewBitmapButton wxNewBitmapButton
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxnewbitmapbutton')")
[K] wxNewBitmapButton
[K] wxNewBitmapButton

$#+K!**wxNewBitmapButton::~wxNewBitmapButton**

~wxNewBitmapButton()K

Destructor.

---

wxNewBitmapButton::~wxNewBitmapButton
wxnewbitmapbuttondtor
browse00563
K wxNewBitmapButton ~wxNewBitmapButton
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxnewbitmapbutton')")
K ~wxNewBitmapButton

$#+K!wxNewBitmapButton::DestroyLabels

void DestroyLabels()[K]

Internal function for destroying labels.

---

[w]xNewBitmapButton::DestroyLabels
[w]xnewbitmapbuttondestroylabels
[b]rowse00564
[K] wxNewBitmapButton DestroyLabels
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxnewbitmapbutton')")
[K] DestroyLabels

$#+K!**wxNewBitmapButton::DrawDecorations**

void DrawDecorations(**wxDC&** *dc*)K

Draws the decorations.

---

wxNewBitmapButton::DrawDecorations
wxnewbitmapbuttondrawdecorations
browse00565
K wxNewBitmapButton DrawDecorations
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxnewbitmapbutton')")
K DrawDecorations

$#+K!**wxNewBitmapButton::DrawLabel**

void DrawLabel(**wxDC&** *dc*)K

Draws the label.

---

wxNewBitmapButton::DrawLabel
wxnewbitmapbuttondrawlabel
browse00566
K wxNewBitmapButton DrawLabel
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxnewbitmapbutton')")
K DrawLabel

$#+K!**wxNewBitmapButton::DrawShade**

void DrawShade(**int** *outerLevel*, **wxDC&** *dc*, **wxPen&** *upperLeftSidePen*, **wxPen&** *lowerRightSidePen*)K

Draws shading on the button.

---

wxNewBitmapButton::DrawShade
wxnewbitmapbuttondrawshade
browse00567
K wxNewBitmapButton DrawShade
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxnewbitmapbutton')")
K DrawShade

$#+K!**wxNewBitmapButton::GetStateLabel**

wxBitmap* GetStateLabel()K

Returns the label that matches the current button state.

---

wxNewBitmapButton::GetStateLabel
wxnewbitmapbuttongetstatelabel
browse00568
K wxNewBitmapButton GetStateLabel
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxnewbitmapbutton')")
K GetStateLabel

$#+K!wxNewBitmapButton::IsInWindow

bool IsInWindow(**int** *x*, **int** *y*)[K]

Returns TRUE if the given point is in the window.

---

[w]xNewBitmapButton::IsInWindow
[w]xnewbitmapbuttonisinwindow
[b]rowse00569
[K] wxNewBitmapButton IsInWindow
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxnewbitmapbutton')")
[K] IsInWindow

$#+K!**wxNewBitmapButton::OnEraseBackground**

void OnEraseBackground(**wxEraseEvent&** *event*)K

Responds to an erase background event.

---

wxNewBitmapButton::OnEraseBackground
wxnewbitmapbuttononerasebackground
browse00570
K wxNewBitmapButton OnEraseBackground
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxnewbitmapbutton')")
K OnEraseBackground

$#+K!**wxNewBitmapButton::OnKillFocus**

void OnKillFocus(**wxFocusEvent&** *event*)K

Responds to a kill focus event.

---

wxNewBitmapButton::OnKillFocus
wxnewbitmapbuttononkillfocus
browse00571
K wxNewBitmapButton OnKillFocus
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxnewbitmapbutton')")
K OnKillFocus

$#+K!wxNewBitmapButton::OnLButtonDown

void OnLButtonDown(**wxMouseEvent&** *event*)K

Responds to a left mouse button down event.

---

wxNewBitmapButton::OnLButtonDown
wxnewbitmapbuttononlbuttondown
browse00572
K wxNewBitmapButton OnLButtonDown
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxnewbitmapbutton')")
K OnLButtonDown

$#+K!**wxNewBitmapButton::OnLButtonUp**

void OnLButtonUp(**wxMouseEvent&** *event*)K

Responds to a left mouse button up event.

---

wxNewBitmapButton::OnLButtonUp
wxnewbitmapbuttononlbuttonup
browse00573
K wxNewBitmapButton OnLButtonUp
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxnewbitmapbutton')")
K OnLButtonUp

$#+K!**wxNewBitmapButton::OnMouseMove**

void OnMouseMove(**wxMouseEvent&** *event*)K

Responds to a mouse move event.

---

wxNewBitmapButton::OnMouseMove
wxnewbitmapbuttononmousemove
browse00574
K wxNewBitmapButton OnMouseMove
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxnewbitmapbutton')")
K OnMouseMove

$#+K!**wxNewBitmapButton::OnPaint**

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

Responds to a paint event.

---

wxNewBitmapButton::OnPaint
wxnewbitmapbuttononpaint
browse00575
K wxNewBitmapButton OnPaint
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxnewbitmapbutton')")
K OnPaint

$#+K!wxNewBitmapButton::OnSize

void OnSize(**wxSizeEvent&** *event*)K

Responds to a size event.

---

wxNewBitmapButton::OnSize
wxnewbitmapbuttononsize
browse00576
K wxNewBitmapButton OnSize
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxnewbitmapbutton')")
K OnSize

$#+K!**wxNewBitmapButton::RenderAllLabelImages**

void RenderAllLabelImages()K

Renders label images.

---

wxNewBitmapButton::RenderAllLabelImages
wxnewbitmapbuttonrenderalllabelimages
browse00577
K wxNewBitmapButton RenderAllLabelImages
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxnewbitmapbutton')")
K RenderAllLabelImages

$#+K!wxNewBitmapButton::RenderLabelImage

void RenderLabelImage(**wxBitmap*&** *destBmp*, **wxBitmap*** *srcBmp*, **bool** *isEnabled = TRUE*, **bool** *isPressed = FALSE*)K

Renders the label image.

---

wxNewBitmapButton::RenderLabelImage
wxnewbitmapbuttonrenderlabelimage
browse00578
K wxNewBitmapButton RenderLabelImage
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxnewbitmapbutton')")
K RenderLabelImage

$#+K!wxNewBitmapButton::RenderLabelImages

void RenderLabelImages()[K]

Renders label images.

---

[w]xNewBitmapButton::RenderLabelImages
[w]xnewbitmapbuttonrenderlabelimages
[b]rowse00579
[K] wxNewBitmapButton RenderLabelImages
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxnewbitmapbutton')")
[K] RenderLabelImages

$#+K!**wxNewBitmapButton::Reshape**

void Reshape()K

This function should be called after Create. It renders the labels, having reloaded the button image if necessary.

---

wxNewBitmapButton::Reshape
wxnewbitmapbuttonreshape
browse00580
K wxNewBitmapButton Reshape
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxnewbitmapbutton')")
K Reshape

$#+K!**wxNewBitmapButton::SetAlignments**

void SetAlignments(**int** *alignText = NB_ALIGN_TEXT_BOTTOM*, **int** *marginX = 2*, **int** *marginY = 2*, **int** *textToLabelGap = 2*)[K]

Sets the text alignment and margins.

---

[w]xNewBitmapButton::SetAlignments
[w]xnewbitmapbuttonsetalignments
[b]rowse00581
[K] wxNewBitmapButton SetAlignments
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxnewbitmapbutton')")
[K] SetAlignments

$#+K!**wxNewBitmapButton::SetLabel**

void SetLabel(**const wxBitmap&** *labelBitmap*, **const wxString&** *labelText = ""*)K

Sets the label and optionally label text.

---

wxNewBitmapButton::SetLabel
wxnewbitmapbuttonsetlabel
browse00582
K wxNewBitmapButton SetLabel
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxnewbitmapbutton')")
K SetLabel

$#+K!**wxToolWindow::wxToolWindow**

wxToolWindow()K

Default constructor.

---

wxToolWindow::wxToolWindow
wxtoolwindowwxtoolwindow
browse00585
K wxToolWindow wxToolWindow
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxtoolwindow')")
K wxToolWindow

$#+K!**wxToolWindow::~wxToolWindow**

~wxToolWindow()K

Destructor.

---

wxToolWindow::~wxToolWindow
wxtoolwindowdtor
browse00586
K wxToolWindow ~wxToolWindow
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxtoolwindow')")
K ~wxToolWindow

$#+K!**wxToolWindow::AddMiniButton**

void AddMiniButton(**cbMiniButton*** *pBtn*)K

Adds a button. Buttons are added in right-to-left order.

---

wxToolWindow::AddMiniButton
wxtoolwindowaddminibutton
browse00587
K wxToolWindow AddMiniButton
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxtoolwindow')")
K AddMiniButton

$#+K!**wxToolWindow::AdjustRectPos**

void AdjustRectPos(**const wxRect&** *original*, **const wxSize&** *newDim*, **wxRect&** *newRect*)K

Helper function.

---

wxToolWindow::AdjustRectPos
wxtoolwindowadjustrectpos
browse00588
K wxToolWindow AdjustRectPos
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxtoolwindow')")
K AdjustRectPos

$#+K!**wxToolWindow::CalcResizedRect**

void CalcResizedRect(**wxRect&** *rect*, **wxPoint&** *delta*, **const wxSize&** *minDim*)K

Calculate resized rectangle.

---

wxToolWindow::CalcResizedRect
wxtoolwindowcalcresizedrect
browse00589
K wxToolWindow CalcResizedRect
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxtoolwindow')")
K CalcResizedRect

$#+K!**wxToolWindow::DrawHintRect**

void DrawHintRect(**const wxRect&** *r*)K

Draws the hint rectangle.

---

wxToolWindow::DrawHintRect
wxtoolwindowdrawhintrect
browse00590
K wxToolWindow DrawHintRect
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxtoolwindow')")
K DrawHintRect

$#+K!**wxToolWindow::GetClient**

wxWindow* GetClient()K

Returns the client window.

---

wxToolWindow::GetClient
wxtoolwindowgetclient
browse00591
K wxToolWindow GetClient
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxtoolwindow')")
K GetClient

$#+K!**wxToolWindow::GetMinimalWndDim**

wxSize GetMinimalWndDim()K

Helper function.

---

wxToolWindow::GetMinimalWndDim
wxtoolwindowgetminimalwnddim
browse00592
K wxToolWindow GetMinimalWndDim
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxtoolwindow')")
K GetMinimalWndDim

$#+K!**wxToolWindow::GetPreferredSize**

wxSize GetPreferredSize(**const wxSize&** *given*)K

Returns the preferred size for the window.

---

wxToolWindow::GetPreferredSize
wxtoolwindowgetpreferredsize
browse00593
K wxToolWindow GetPreferredSize
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxtoolwindow')")
K GetPreferredSize

$#+K!**wxToolWindow::GetScrMousePos**

void GetScrMousePos(**wxMouseEvent&** *event*, **wxPoint&** *pos*)K

Gets the mouse position in screen coordinates.

---

wxToolWindow::GetScrMousePos
wxtoolwindowgetscrmousepos
browse00594
K wxToolWindow GetScrMousePos
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxtoolwindow')")
K GetScrMousePos

$#+K!**wxToolWindow::GetScrWindowRect**

void GetScrWindowRect(**wxRect&** *r*)K

Maps client coordinates to screen coordinates.

---

wxToolWindow::GetScrWindowRect
wxtoolwindowgetscrwindowrect
browse00595
K wxToolWindow GetScrWindowRect
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxtoolwindow')")
K GetScrWindowRect

$#+K!**wxToolWindow::HandleTitleClick**

bool HandleTitleClick(**wxMouseEvent&** *event*)K

Handles clicking on the title. By default, does nothing.

---

wxToolWindow::HandleTitleClick
wxtoolwindowhandletitleclick
browse00596
K wxToolWindow HandleTitleClick
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxtoolwindow')")
K HandleTitleClick

$#+K!**wxToolWindow::HitTestWindow**

int HitTestWindow(**wxMouseEvent&** *event*)K

Tests if the mouse position is in this window.

---

wxToolWindow::HitTestWindow
wxtoolwindowhittestwindow
browse00597
K wxToolWindow HitTestWindow
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxtoolwindow')")
K HitTestWindow

$#+K!**wxToolWindow::LayoutMiniButtons**

void LayoutMiniButtons()K

Lays out the buttons.

---

wxToolWindow::LayoutMiniButtons
wxtoolwindowlayoutminibuttons
browse00598
K wxToolWindow LayoutMiniButtons
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxtoolwindow')")
K LayoutMiniButtons

$#+K!**wxToolWindow::OnEraseBackground**

void OnEraseBackground(**wxEraseEvent&** *event*)K

Responds to an erase background event.

---

wxToolWindow::OnEraseBackground
wxtoolwindowonerasebackground
browse00599
K wxToolWindow OnEraseBackground
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxtoolwindow')")
K OnEraseBackground

$#+K!**wxToolWindow::OnLeftDown**

void OnLeftDown(**wxMouseEvent&** *event*)K

Responds to a mouse left down event.

---

wxToolWindow::OnLeftDown
wxtoolwindowonleftdown
browse00600
K wxToolWindow OnLeftDown
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxtoolwindow')")
K OnLeftDown

$#+K!**wxToolWindow::OnLeftUp**

void OnLeftUp(**wxMouseEvent&** *event*)K

Responds to a mouse left up event.

---

wxToolWindow::OnLeftUp
wxtoolwindowonleftup
browse00601
K wxToolWindow OnLeftUp
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxtoolwindow')")
K OnLeftUp

$#+K! wxToolWindow::OnMiniButtonClicked

void OnMiniButtonClicked(**int** *btnIdx*)^K

Called when a mini button is clicked. By default, does nothing.

---

^w xToolWindow::OnMiniButtonClicked
^w xtoolwindowonminibuttonclicked
^b rowse00602
^K wxToolWindow OnMiniButtonClicked
^E nableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxtoolwindow')")
^K OnMiniButtonClicked

$#+K!**wxToolWindow::OnMotion**

void OnMotion(**wxMouseEvent&** *event*)K

Responds to a mouse move event.

---

wxToolWindow::OnMotion
wxtoolwindowonmotion
browse00603
K wxToolWindow OnMotion
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxtoolwindow')")
K OnMotion

$#+K!**wxToolWindow::OnPaint**

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

Responds to a paint event.

---

wxToolWindow::OnPaint
wxtoolwindowonpaint
browse00604
K wxToolWindow OnPaint
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxtoolwindow')")
K OnPaint

$#+K!**wxToolWindow::OnSize**

void OnSize(**wxSizeEvent&** *event*)K

Responds to a size event.

---

wxToolWindow::OnSize
wxtoolwindowonsize
browse00605
K wxToolWindow OnSize
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxtoolwindow')")
K OnSize

$#+K!**wxToolWindow::SetClient**

void SetClient(**wxWindow*** *pWnd*)K

Sets the client for this tool window.

---

wxToolWindow::SetClient
wxtoolwindowsetclient
browse00606
K wxToolWindow SetClient
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxtoolwindow')")
K SetClient

$#+K!**wxToolWindow::SetHintCursor**

void SetHintCursor(**int** *type*)K

Sets the hint cursor.

---

wxToolWindow::SetHintCursor
wxtoolwindowsethintcursor
browse00607
K wxToolWindow SetHintCursor
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxtoolwindow')")
K SetHintCursor

$#+K!**wxToolWindow::SetTitleFont**

void SetTitleFont(**wxFont&** *font*)K

Sets the title font.

---

wxToolWindow::SetTitleFont
wxtoolwindowsettitlefont
browse00608
K wxToolWindow SetTitleFont
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `wxtoolwindow')")
K SetTitleFont

$#+K!**A row of all non-fixed bars don't position properly**

By Julian Smart.

I found that if I added all non-fixed bars, bars would overlap. This seems to be because the proportional resizing doesn't work before the window is laid out. I worked around this by setting pane sizes *before* the bars are added:

```
wxSize sz = GetClientSize();

// Set width for panes to help it do the calculations
int i;
for (i = 0; i < 2; i++)
{
    cbDockPane& pane = * (m_frameLayout->GetPane(i));
    pane.SetPaneWidth(sz.x);
}
```

A row of all non-fixed bars don't position properly
topic3
browse00614
K A row of all non-fixed bars don't position properly
EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`fl.hlp', `faq')")