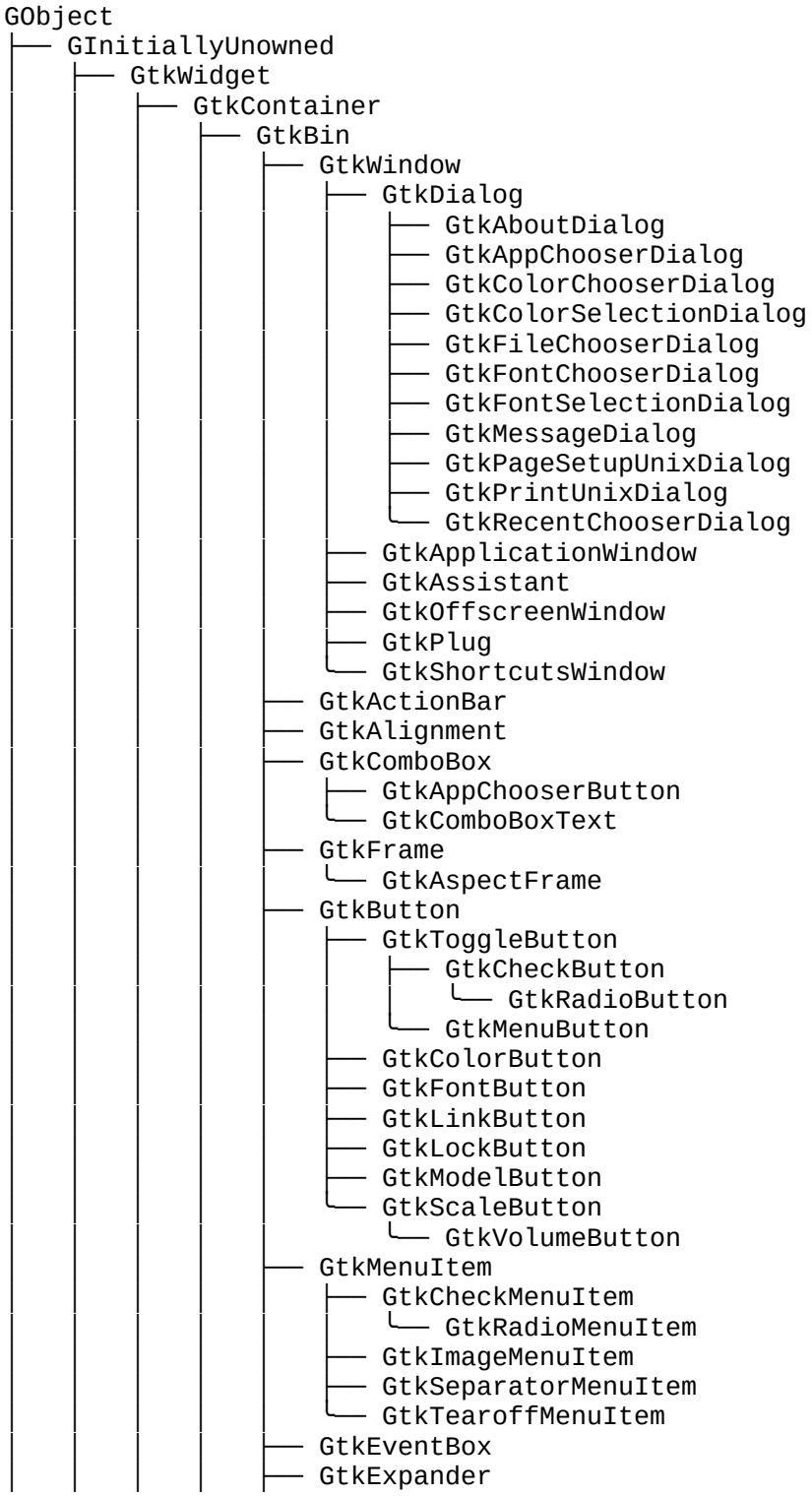
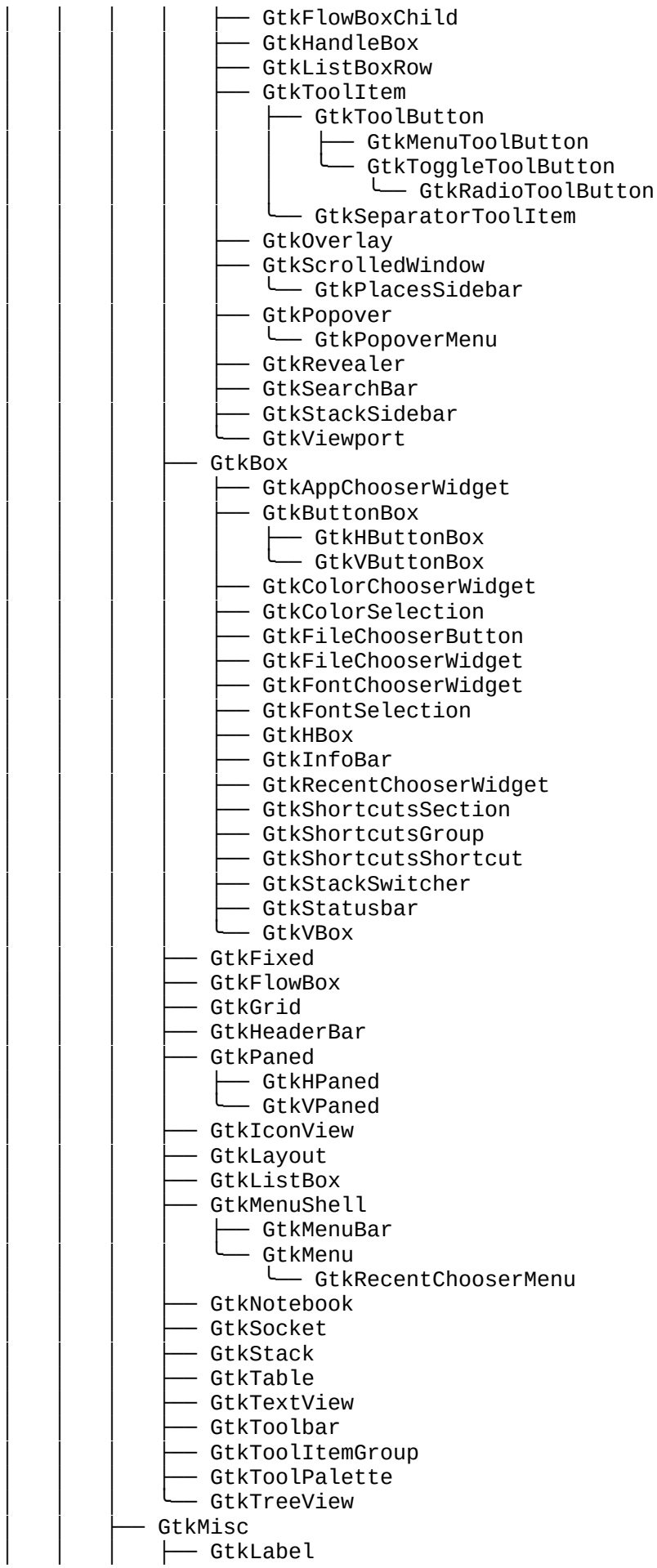
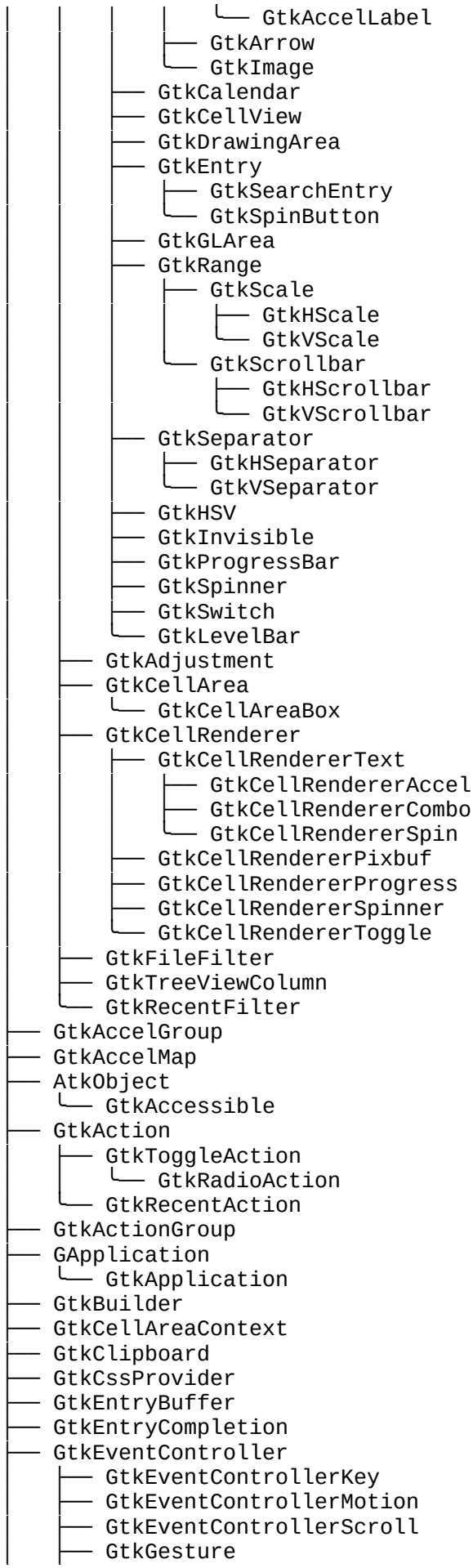
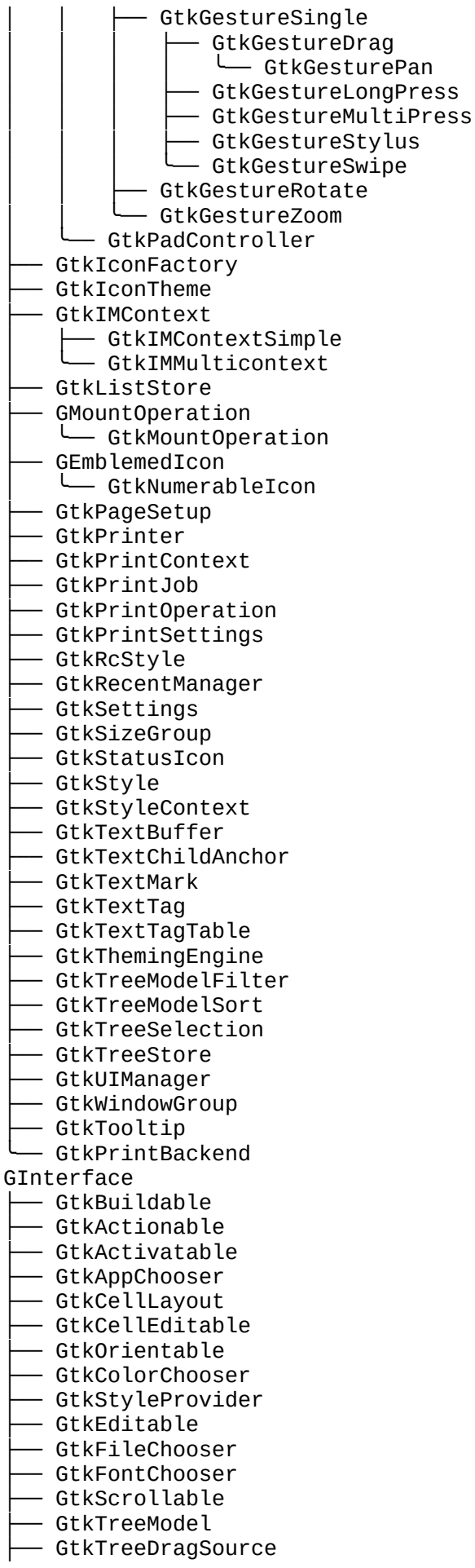
Part II. GTK+ Widgets and Objects: GTK+ 3 Reference Manual

Object Hierarchy




GtkFlowBoxChild
GtkHandleBox
GtkListBoxRow
GtkToolItem
GtkToolButton
GtkMenuToolButton
GtkToggleToolButton
GtkRadioToolButton
GtkSeparatorToolItem
GtkOverlay
GtkScrolledWindow
GtkPlacesSidebar
GtkPopover
GtkPopoverMenu
GtkRevealer
GtkSearchBar
GtkStackSidebar
GtkViewport
GtkBox
GtkAppChooserWidget
GtkButtonBox
GtkHButtonBox
GtkVButtonBox
GtkColorChooserWidget
GtkColorSelection
GtkFileChooserButton
GtkFileChooserWidget
GtkFontChooserWidget
GtkFontSelection
GtkHBox
GtkInfoBar
GtkRecentChooserWidget
GtkShortcutsSection
GtkShortcutsGroup
GtkShortcutsShortcut
GtkStackSwitcher
GtkStatusbar
GtkVBox
GtkFixed
GtkFlowBox
GtkGrid
GtkHeaderBar
GtkPaned
GtkHPaned
GtkVPaned
GtkIconView
GtkLayout
GtkListBox
GtkMenuShell
GtkMenuBar
GtkMenu
GtkRecentChooserMenu
GtkNotebook
GtkSocket
GtkStack
GtkTable
GtkTextView
GtkToolbar
GtkToolItemGroup
GtkToolPalette
GtkTreeView
GtkMisc
GtkLabel

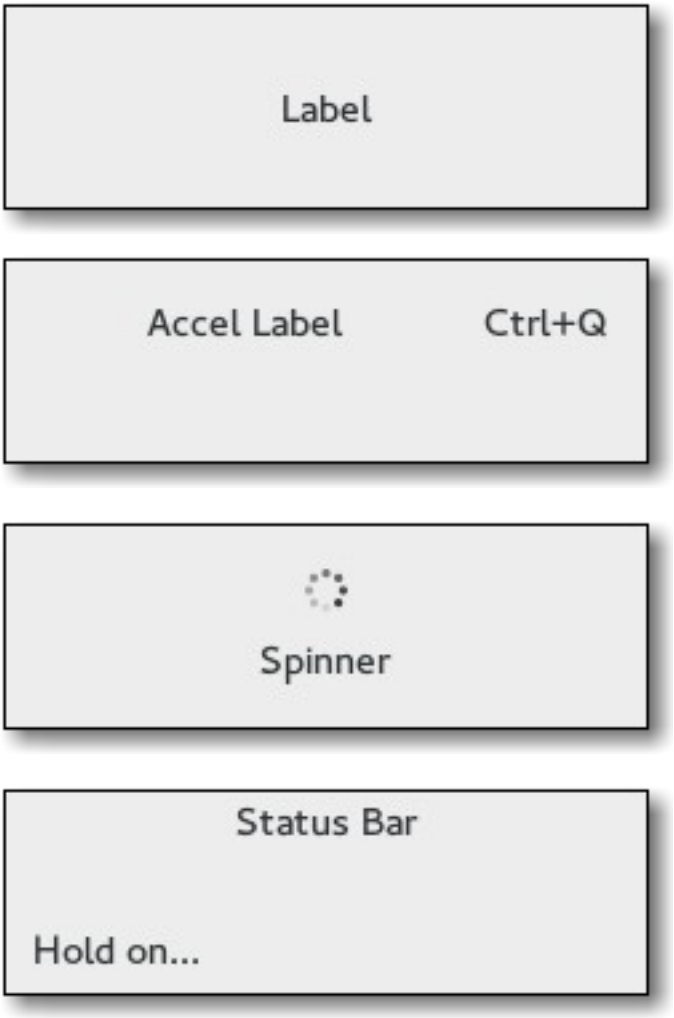

GtkAccelLabel
GtkArrow
GtkImage
GtkCalendar
GtkCellView
GtkDrawingArea
GtkEntry
GtkSearchEntry
GtkSpinButton
GtkGLArea
GtkRange
GtkScale
GtkHScale
GtkVScale
GtkScrollbar
GtkHScrollbar
GtkVScrollbar
GtkSeparator
GtkHSeparator
GtkVSeparator
GtkHSV
GtkInvisible
GtkProgressBar
GtkSpinner
GtkSwitch
GtkLevelBar
GtkAdjustment
GtkCellArea
GtkCellAreaBox
GtkCellRenderer
GtkCellRendererText
GtkCellRendererAccel
GtkCellRendererCombo
GtkCellRendererSpin
GtkCellRendererPixbuf
GtkCellRendererProgress
GtkCellRendererSpinner
GtkCellRendererToggle
GtkFileFilter
GtkTreeViewColumn
GtkRecentFilter
GtkAccelGroup
GtkAccelMap
AtkObject
GtkAccessible
GtkAction
GtkToggleAction
GtkRadioAction
GtkRecentAction
GtkActionGroup
GApplication
GtkApplication
GtkBuilder
GtkCellAreaContext
GtkClipboard
GtkCssProvider
GtkEntryBuffer
GtkEntryCompletion
GtkEventController
GtkEventControllerKey
GtkEventControllerMotion
GtkEventControllerScroll
GtkGesture

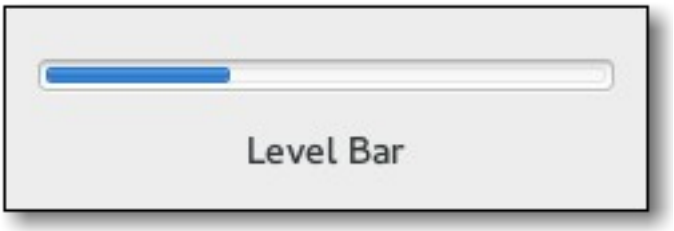

GtkGestureSingle
GtkGestureDrag
GtkGesturePan
GtkGestureLongPress
GtkGestureMultiPress
GtkGestureStylus
GtkGestureSwipe
GtkGestureRotate
GtkGestureZoom
GtkPadController
GtkIconFactory
GtkIconTheme
GtkIMContext
GtkIMContextSimple
GtkIMMulticontext
GtkListStore
GMountOperation
GtkMountOperation
GEmblemedIcon
GtkNumerableIcon
GtkPageSetup
GtkPrinter
GtkPrintContext
GtkPrintJob
GtkPrintOperation
GtkPrintSettings
GtkRcStyle
GtkRecentManager
GtkSettings
GtkSizeGroup
GtkStatusIcon
GtkStyle
GtkStyleContext
GtkTextBuffer
GtkTextChildAnchor
GtkTextMark
GtkTextTag
GtkTextTagTable
GtkThemingEngine
GtkTreeModelFilter
GtkTreeModelSort
GtkTreeSelection
GtkTreeStore
GtkUIManager
GtkWindowGroup
GtkTooltip
GtkPrintBackend
GInterface
GtkBuildable
GtkActionable
GtkActivatable
GtkAppChooser
GtkCellLayout
GtkCellEditable
GtkOrientable
GtkColorChooser
GtkStyleProvider
GtkEditable
GtkFileChooser
GtkFontChooser
GtkScrollable
GtkTreeModel
GtkTreeDragSource

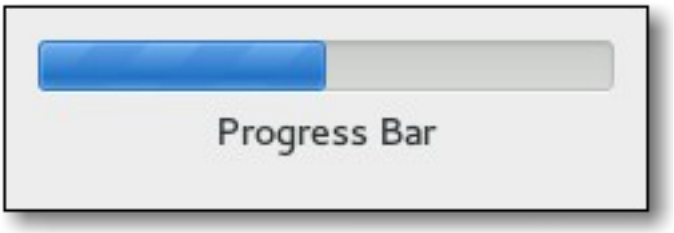
```
    ├── GtkTreeDragDest
    ├── GtkTreeSortable
    ├── GtkPrintOperationPreview
    ├── GtkRecentChooser
    └── GtkToolShell
GBoxed
    ├── GtkPaperSize
    ├── GtkTextIter
    ├── GtkSelectionData
    ├── GtkRequisition
    ├── GtkBorder
    ├── GtkTreeIter
    ├── GtkCssSection
    ├── GtkTreePath
    ├── GtkIconSet
    └── GtkTargetList
```

Widget Gallery

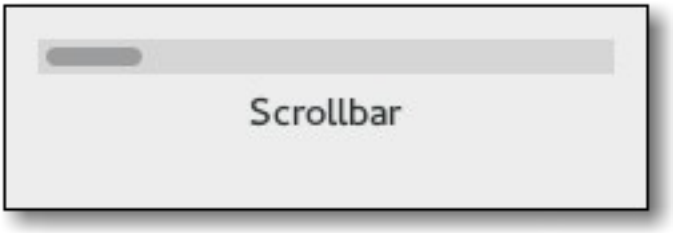
Display



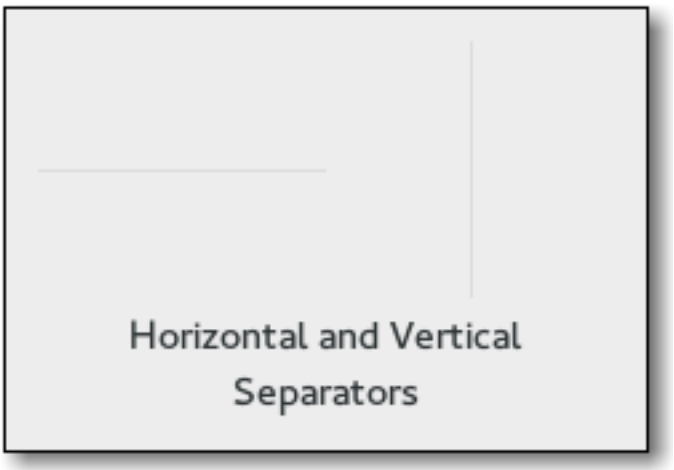

Level Bar

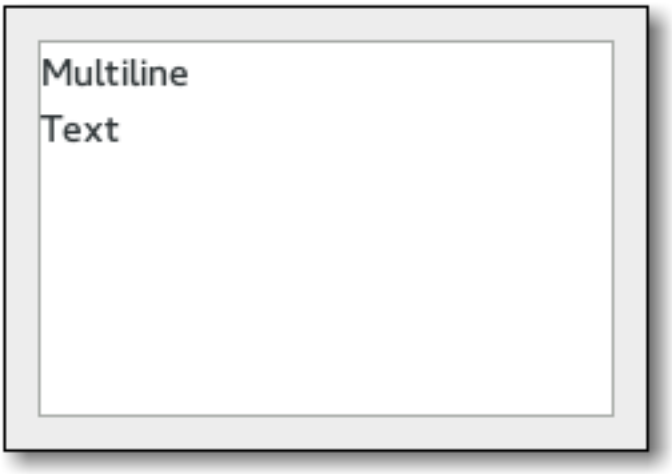

Progress Bar

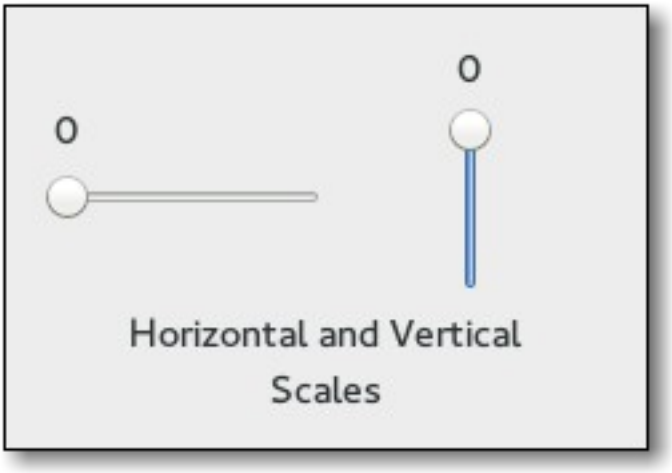

Info Bar


Scrollbar

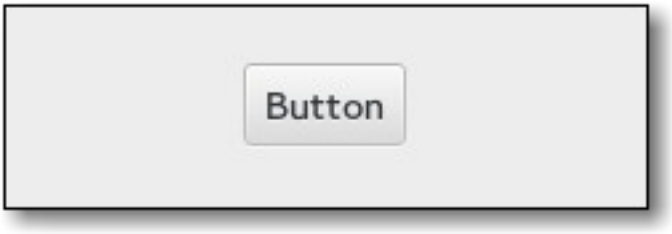

Image

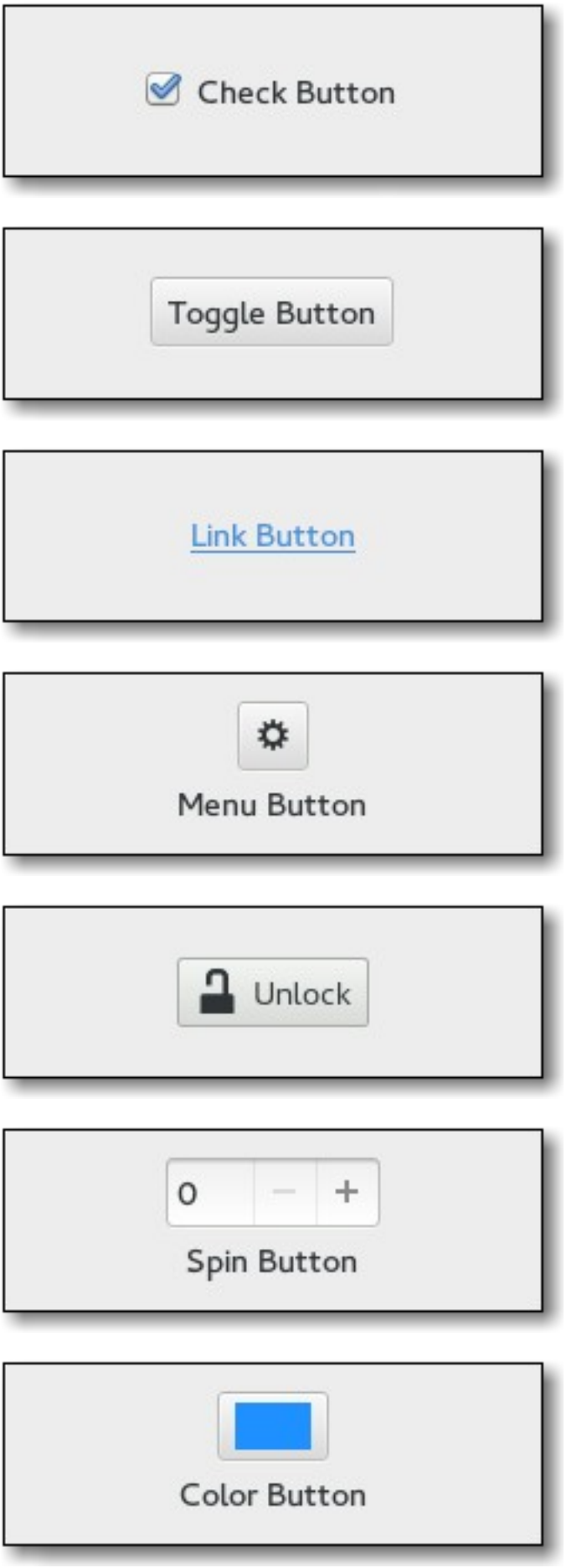

Horizontal and Vertical
Separators





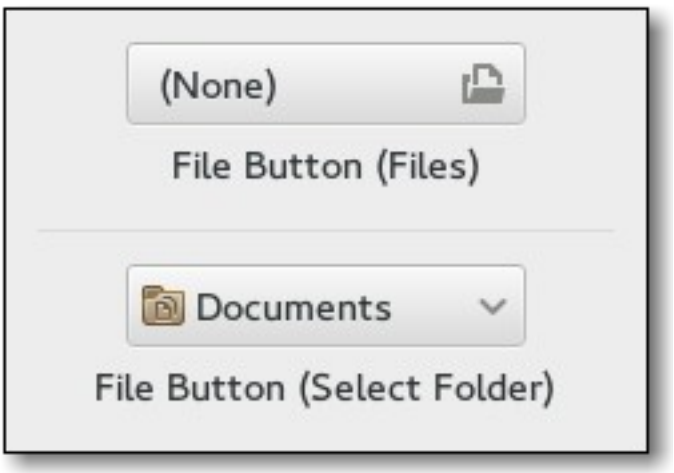
Buttons



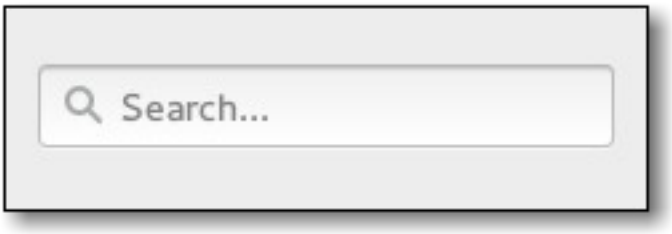

Check Button
Toggle Button
Link Button
Menu Button
Unlock
0
Spin Button
Color Button


ON
OFF
Combo Box
Combo Box Text
Sans Serif 10
Font Button
gedit
Application Button

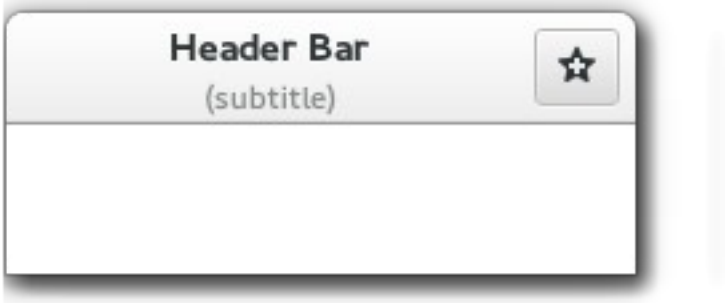

Radio Button One
Radio Button Two
Radio Button Three

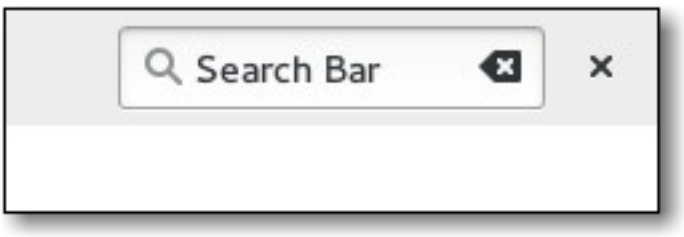

(None)
File Button (Files)
Documents
File Button (Select Folder)

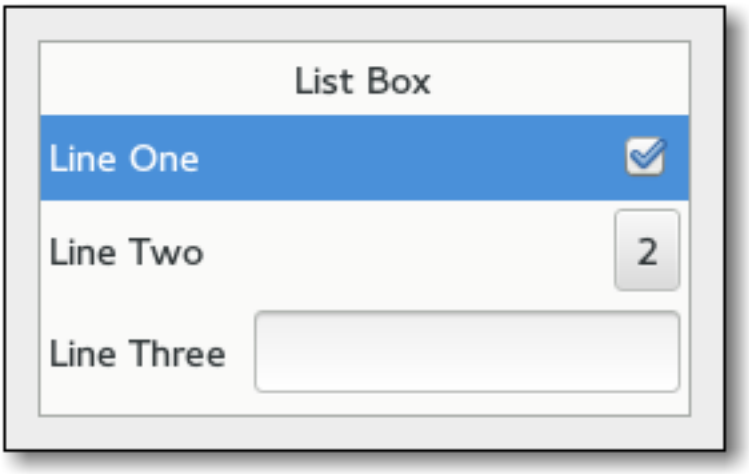
Entries

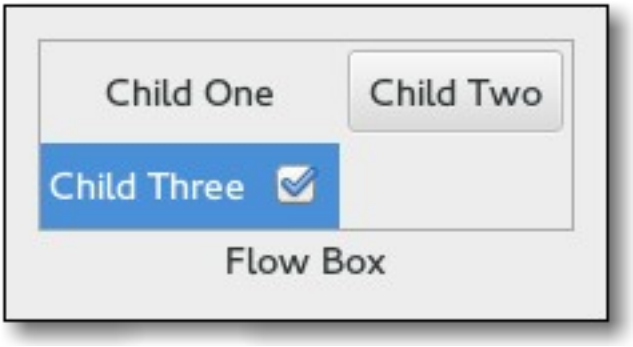


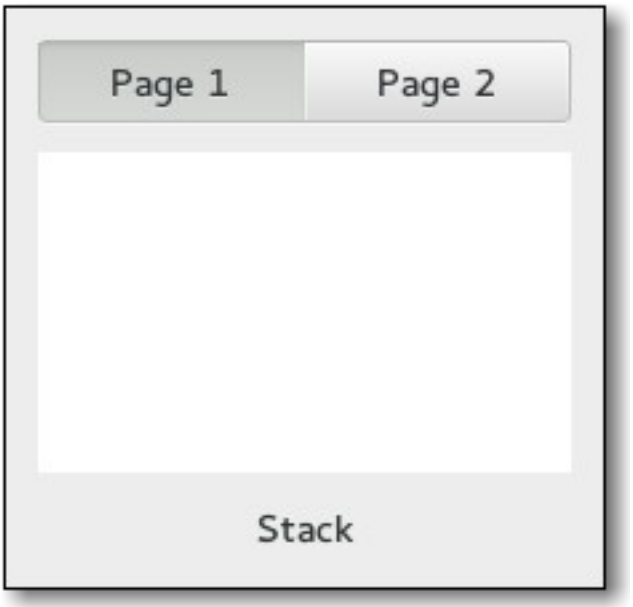
Containers

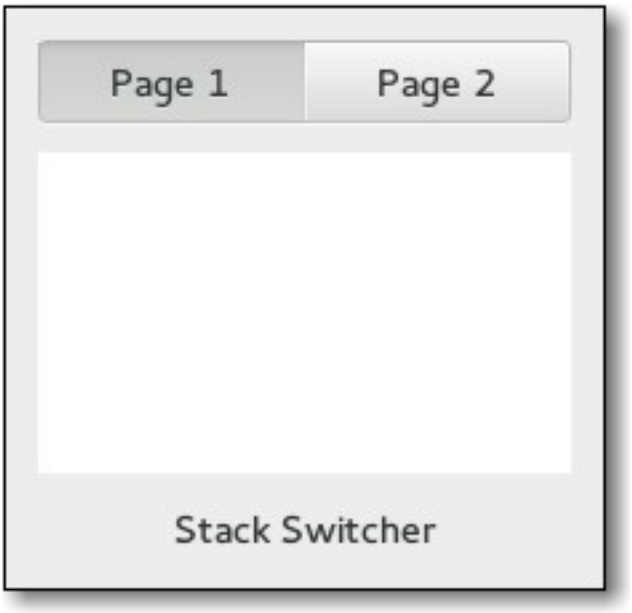


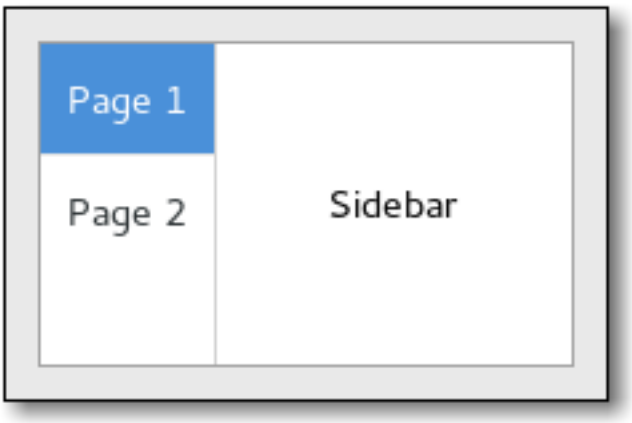


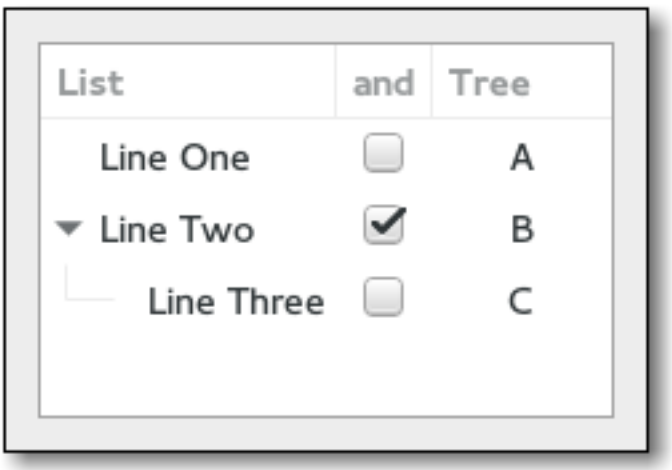

List Box
Line One
Line Two
2
Line Three


Child One
Child Two
Child Three
Flow Box

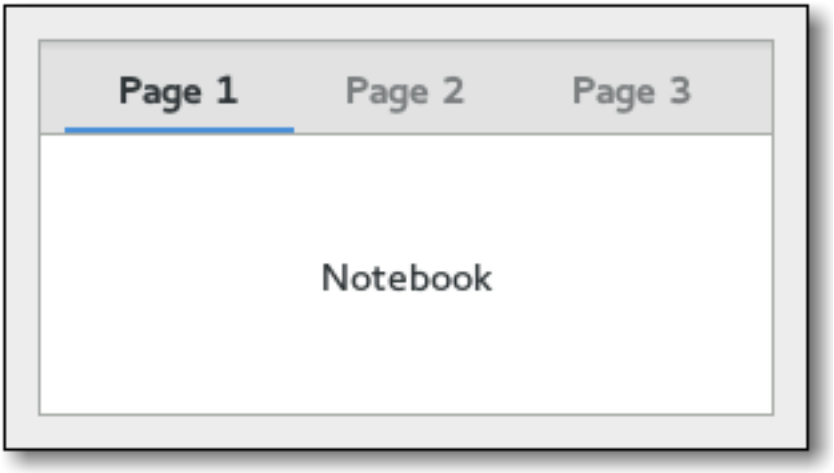

Page 1
Page 2
Stack


Page 1
Page 2
Stack Switcher

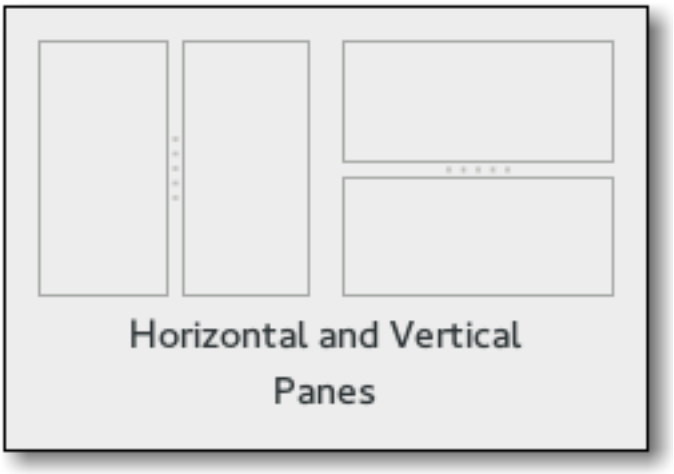

Page 1
Page 2
Sidebar

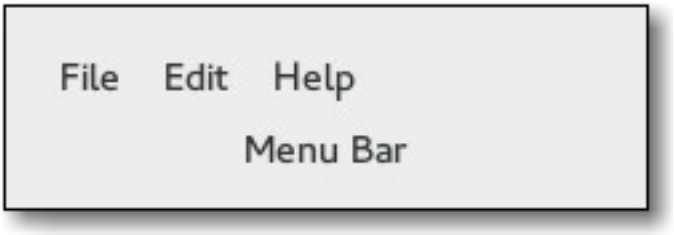

List
and
Tree
Line One
A
Line Two
B
Line Three
C

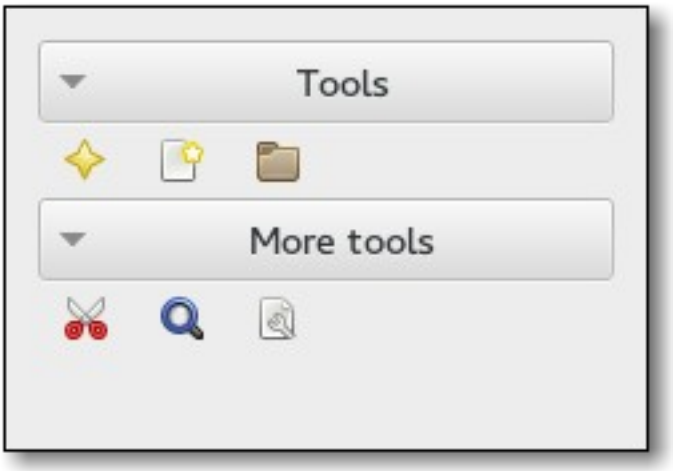

One
Two
Icon View

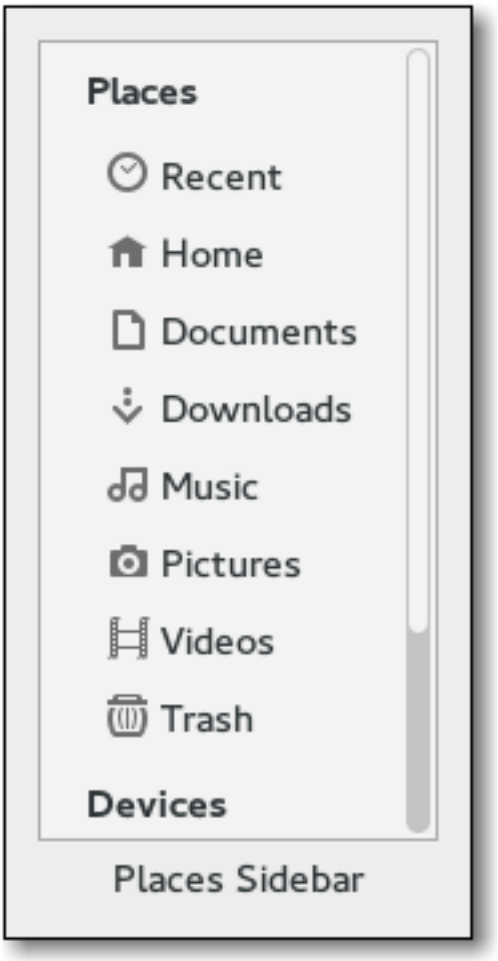

Page 1
Page 2
Page 3
Notebook


Frame

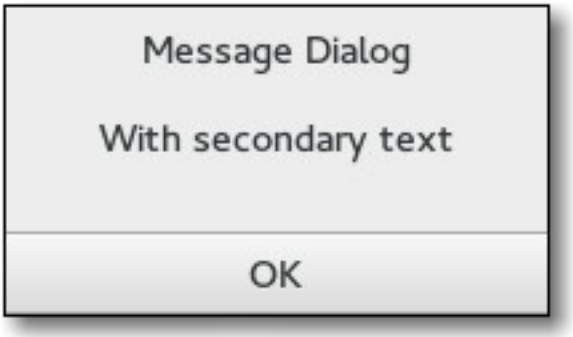

Horizontal and Vertical
Panes


File
Edit
Help
Menu Bar

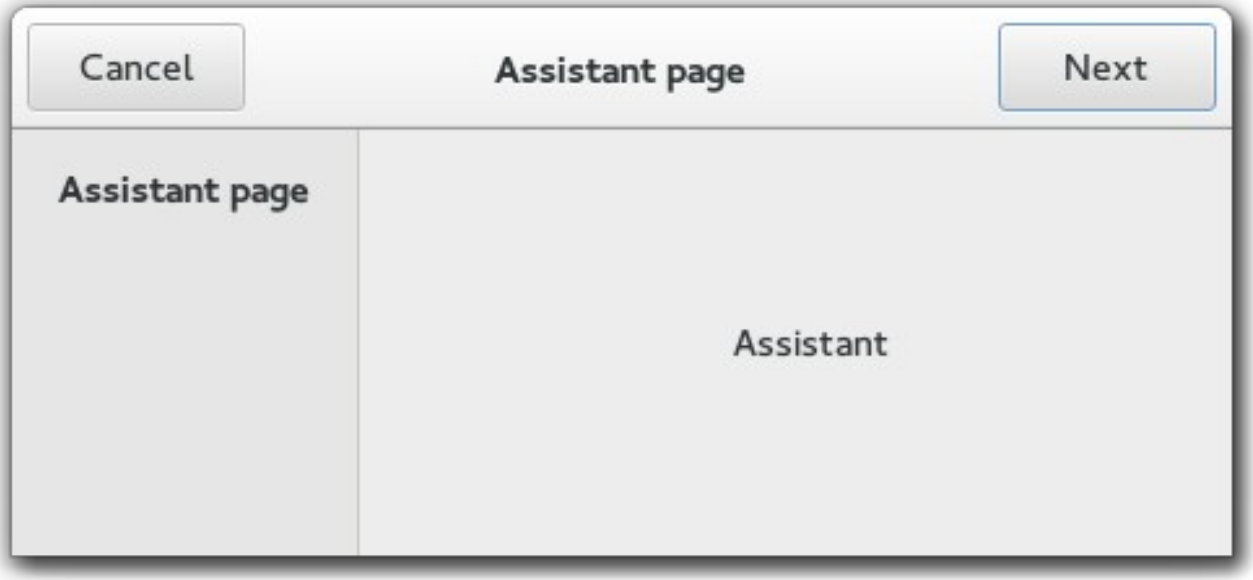

Tools
More tools


Places
Recent
Home
Documents
Downloads
Music
Pictures
Videos
Trash
Devices
Places Sidebar

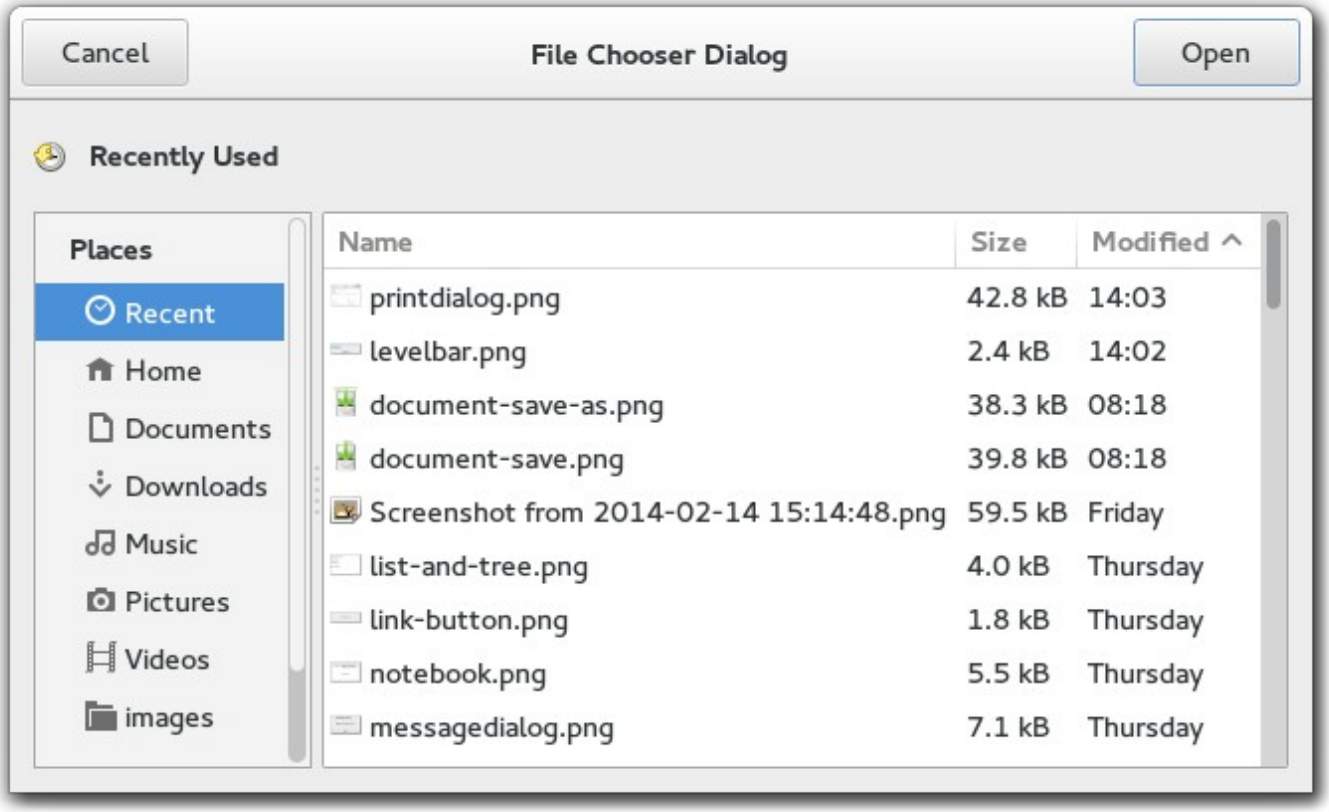
Windows

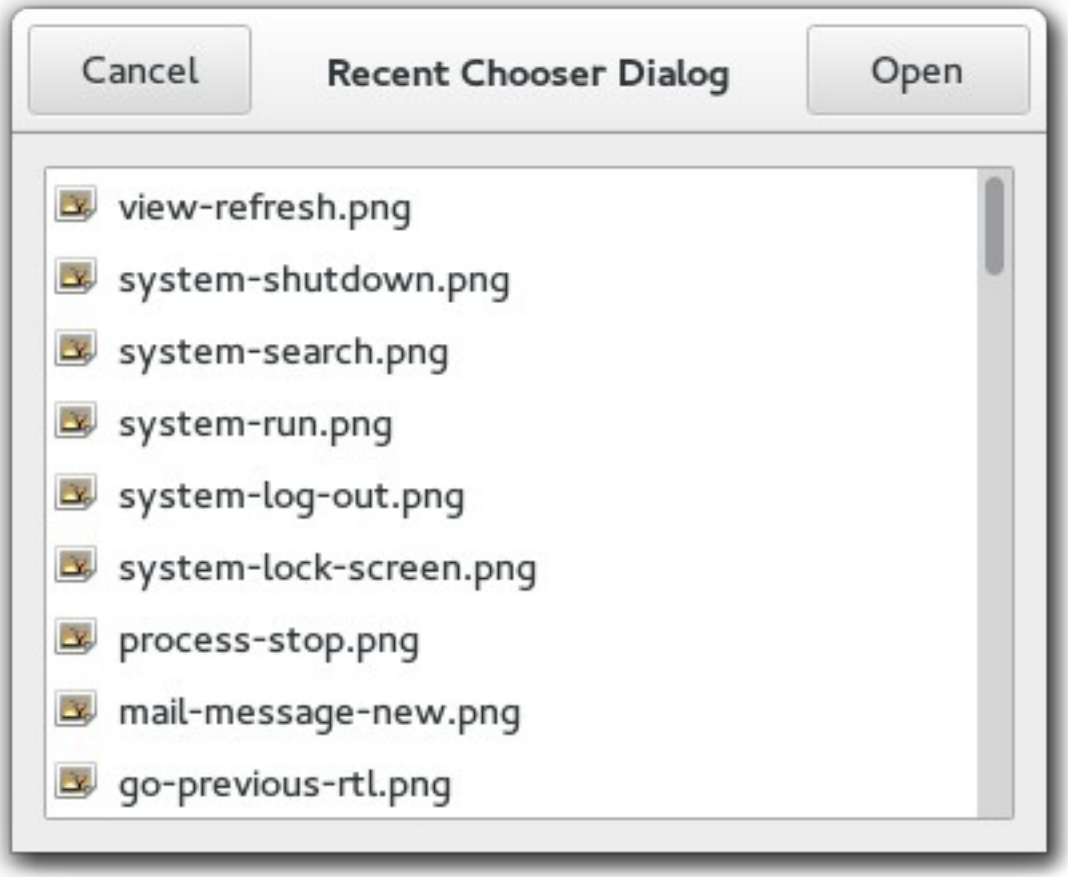

Message Dialog
With secondary text
OK


About
Credits
GTK+ Code Demos
3.11.8
Program to demonstrate GTK+ functions.
Website
© 1997-2013 The GTK+ Team

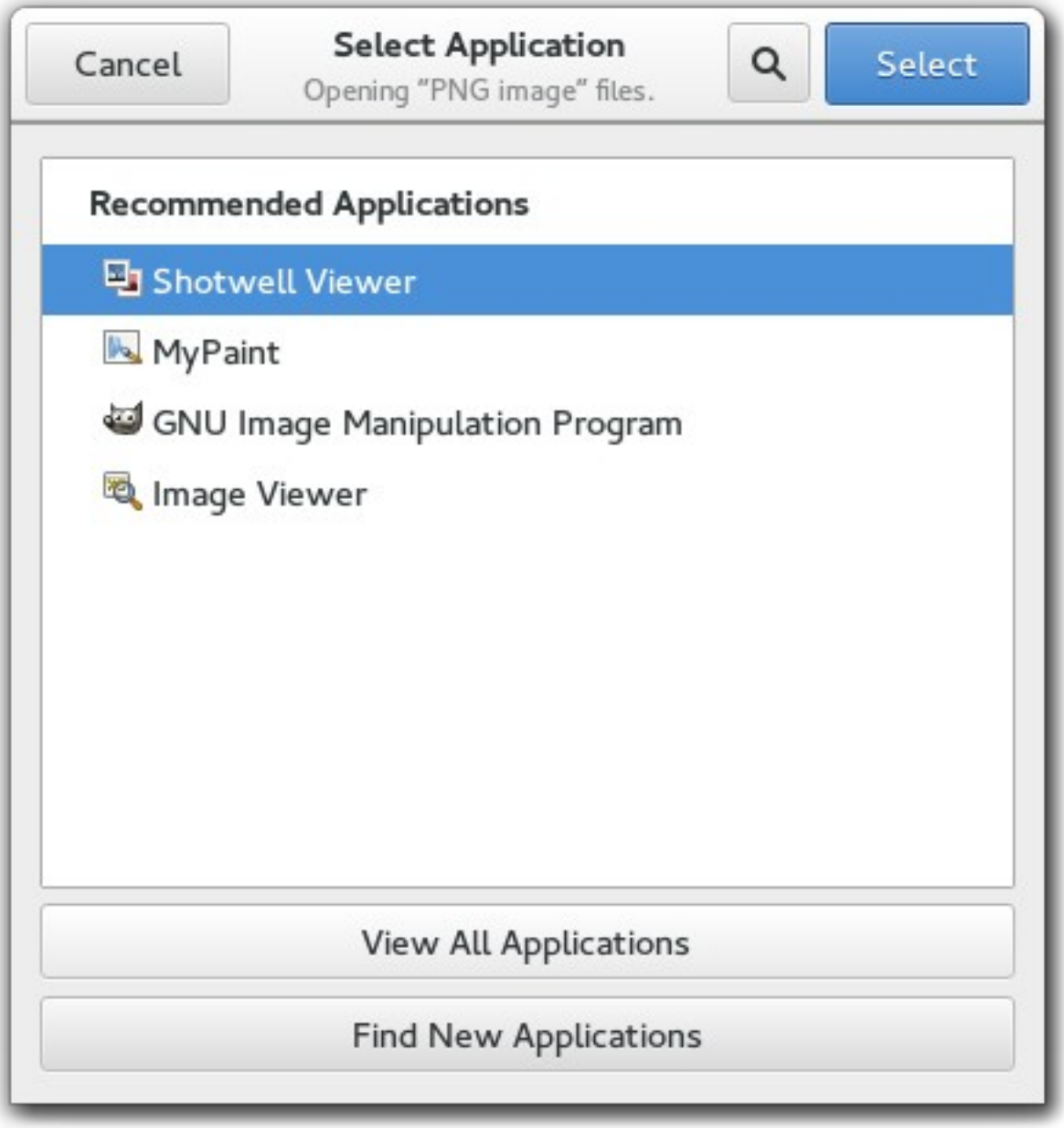

Cancel
Assistant page
Next
Assistant page
Assistant

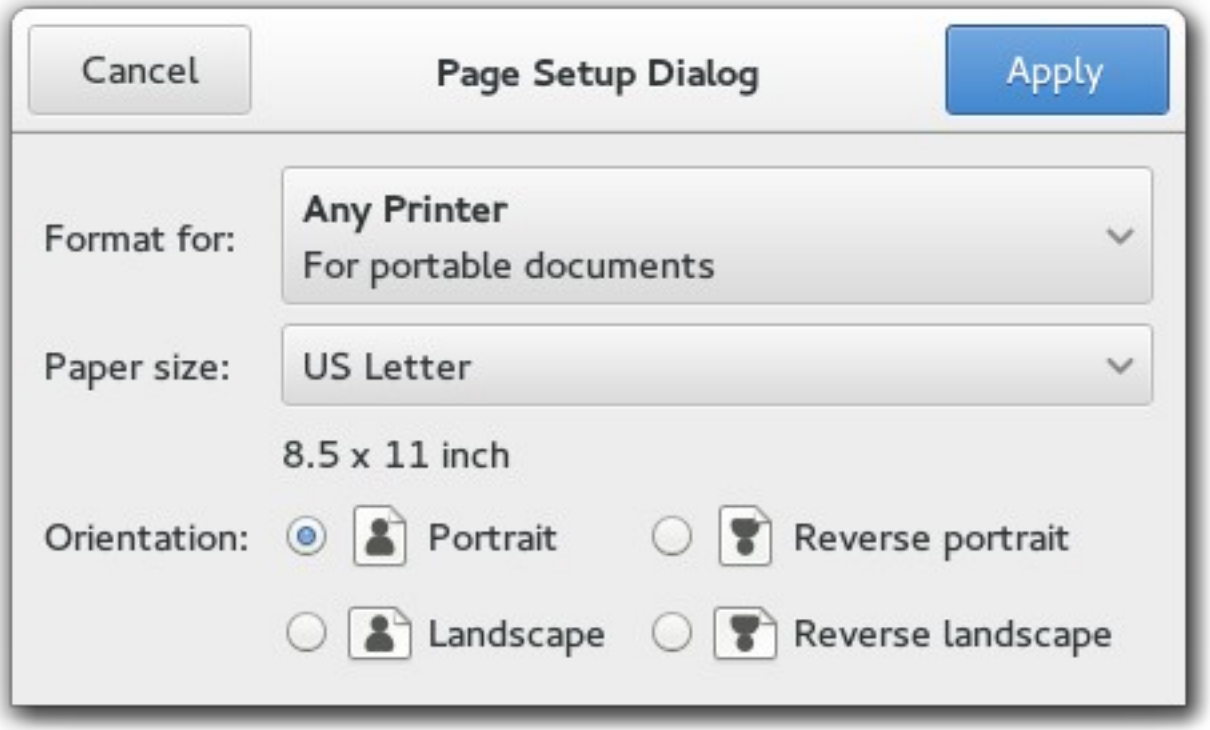

Cancel
Color Chooser Dialog
Select
Custom

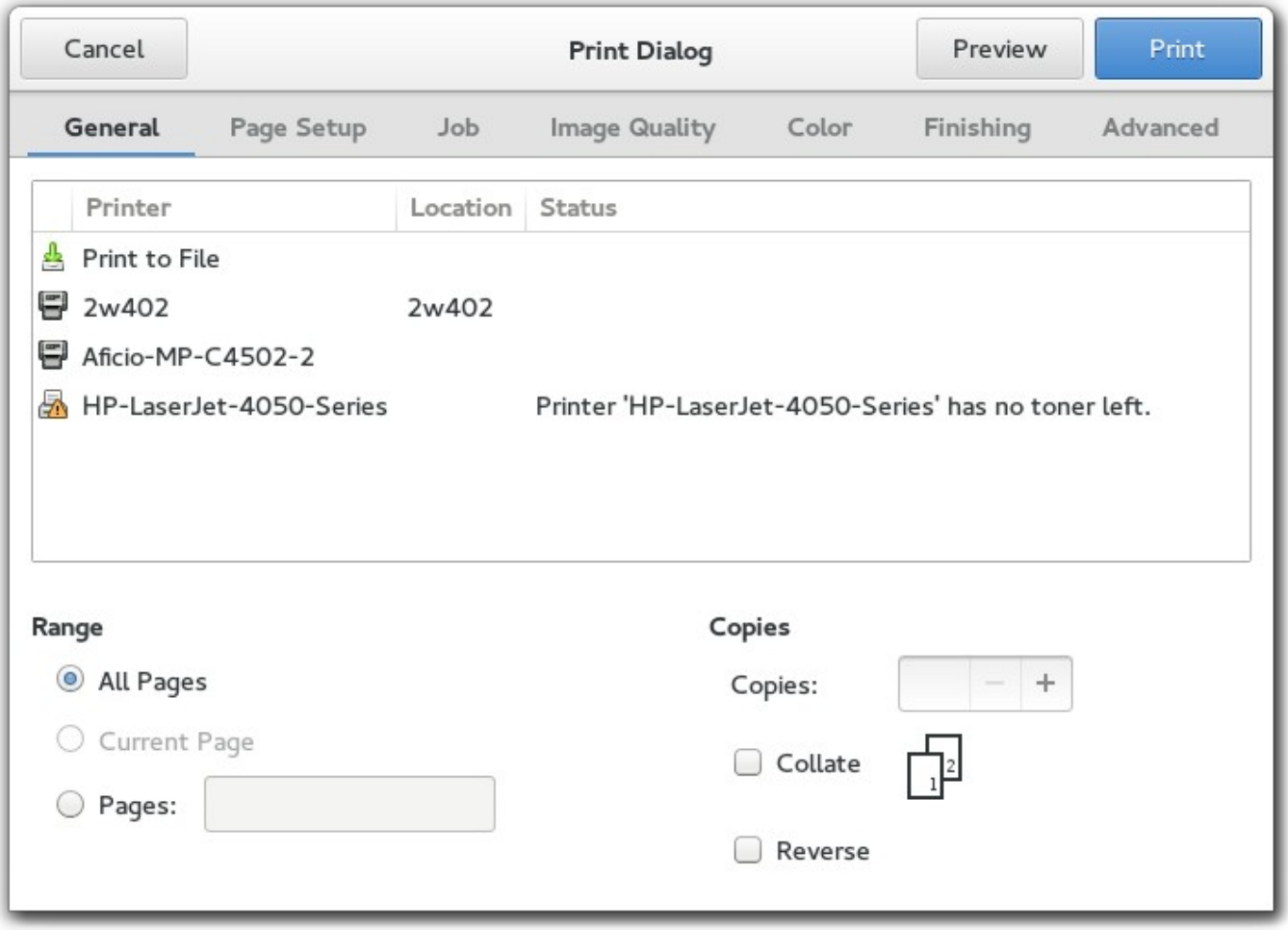

Cancel
File Chooser Dialog
Open
Recently Used
Places
Recent
Home
Documents
Downloads
Music
Pictures
Videos
images
Name
Size
Modified
printdialog.png 42.8 kB 14:03
levelbar.png 2.4 kB 14:02
document-save-as.png 38.3 kB 08:18
document-save.png 39.8 kB 08:18
Screenshot from 2014-02-14 15:14:48.png 59.5 kB Friday
list-and-tree.png 4.0 kB Thursday
link-button.png 1.8 kB Thursday
notebook.png 5.5 kB Thursday
messagedialog.png 7.1 kB Thursday


Cancel
Recent Chooser Dialog
Open
view-refresh.png
system-shutdown.png
system-search.png
system-run.png
system-log-out.png
system-lock-screen.png
process-stop.png
mail-message-new.png
go-previous-rtl.png


Cancel
Font Chooser Dialog
Select
Search font name
Sans
The quick brown fox jumps over the lazy ...
Sans Bold
The quick brown fox jumps over the l...
Sans Italic
The quick brown fox jumps over the lazy ...
Sans Bold Italic
The quick brown fox jumps over the l...
Serif
The quick brown fox jumps over the lazy ...
Serif Bold
The quick brown fox jumps over the lazy dog.
10


Cancel
Select Application
Opening "PNG image" files.
Select
Recommended Applications
Shotwell Viewer
MyPaint
GNU Image Manipulation Program
Image Viewer
View All Applications
Find New Applications


Cancel
Page Setup Dialog
Apply
Format for:
Any Printer
For portable documents
Paper size:
US Letter
8.5 x 11 inch
Orientation:
Portrait
Reverse portrait
Landscape
Reverse landscape



Application support

GtkApplication — Application class

GtkApplicationWindow — GtkWindow subclass with GtkApplication support

GtkActionable — An interface for widgets that can be associated with actions

---

GtkApplication

GtkApplication — Application class

Functions

GtkApplication *	gtk_application_new ()
void	gtk_application_add_window ()
void	gtk_application_remove_window ()
GList *	gtk_application_get_windows ()
GtkWindow *	gtk_application_get_window_by_id ()
GtkWindow *	gtk_application_get_active_window ()
guint	gtk_application_inhibit ()
void	gtk_application_uninhibit ()
gboolean	gtk_application_is_inhibited ()
gboolean	gtk_application_prefers_app_menu ()
GMenuModel *	gtk_application_get_app_menu ()
void	gtk_application_set_app_menu ()
GMenuModel *	gtk_application_get_menubar ()
void	gtk_application_set_menubar ()
GMenu *	gtk_application_get_menu_by_id ()
void	gtk_application_add_accelerator ()
void	gtk_application_remove_accelerator ()
gchar **	gtk_application_list_action_descriptions ()
gchar **	gtk_application_get_accels_for_action ()
void	gtk_application_set_accels_for_action ()
gchar **	gtk_application_get_actions_for_accel ()

Properties

GtkWindow *	active-window	Read
GMenuModel *	app-menu	Read / Write
GMenuModel *	menubar	Read / Write
gboolean	register-session	Read / Write
gboolean	screensaver-active	Read

Signals

void	query-end	Run First
void	window-added	Run First
void	window-removed	Run First

Types and Values

struct	GtkApplication
struct	GtkApplicationClass
enum	GtkApplicationInhibitFlags

Object Hierarchy

```
GObject
└── GApplication
    └── GtkApplication
```

Implemented Interfaces

GtkApplication implements GActionGroup and GActionMap.

Includes

```
#include <gtk/gtk.h>
```

Description

GtkApplication is a class that handles many important aspects of a GTK+ application in a convenient fashion, without enforcing a one-size-fits-all application model.

Currently, GtkApplication handles GTK+ initialization, application uniqueness, session management, provides some basic scriptability and desktop shell integration by exporting actions and menus and manages a list of toplevel windows whose life-cycle is automatically tied to the life-cycle of your application.

While GtkApplication works fine with plain GtkWindows, it is recommended to use it together with GtkApplicationWindow.

When GDK threads are enabled, GtkApplication will acquire the GDK lock when invoking actions that arrive from other processes. The GDK lock is not touched for local action invocations. In order to have actions invoked in a predictable context it is therefore recommended that the GDK lock be held while invoking actions locally with `g_action_group_activate_action()`. The same applies to actions associated with GtkApplicationWindow and to the “activate” and “open” GApplication methods.

Automatic resources

GtkApplication will automatically load menus from the GtkBuilder resource located at "gtk/menus.ui", relative to the application's resource base path (see `g_application_set_resource_base_path()`). The menu with the ID "app-menu" is taken as the application's app menu and the menu with the ID "menubar" is taken as the application's menubar. Additional menus (most interesting submenus) can be named and accessed via `gtk_application_get_menu_by_id()` which allows for dynamic population of a part of the menu structure.

If the resources "gtk/menus-appmenu.ui" or "gtk/menus-traditional.ui" are present then these files will be used in preference, depending on the value of `gtk_application_prefers_app_menu()`. If the resource "gtk/menus-common.ui" is present it will be loaded as well. This is useful for storing items that are referenced from both "gtk/menus-appmenu.ui" and "gtk/menus-traditional.ui".

It is also possible to provide the menus manually using `gtk_application_set_app_menu()` and `gtk_application_set_menubar()`.

GtkApplication will also automatically setup an icon search path for the default icon theme by appending "icons" to the resource base path. This allows your application to easily store its icons as resources. See `gtk_icon_theme_add_resource_path()` for more information.

If there is a resource located at "gtk/help-overlay.ui" which defines a GtkShortcutsWindow with ID "help_overlay" then GtkApplication associates an instance of this shortcuts window with each GtkApplicationWindow and sets up keyboard accelerators (Control-F1 and Control-?) to open it. To create a menu item that displays the shortcuts window, associate the item with the action win.show-help-overlay.

A simple application

A simple example

GtkApplication optionally registers with a session manager of the users session (if you set the “register-session” property) and offers various functionality related to the session life-cycle.

An application can block various ways to end the session with the `gtk_application_inhibit()` function. Typical use cases for this kind of inhibiting are long-running, uninterruptible operations, such as burning a CD or performing a disk backup. The session manager may not honor the inhibitor, but it can be expected to inform the user about the negative consequences of ending the session while inhibitors are present.

See Also

HowDoI: Using GtkApplication, Getting Started with GTK+: Basics

Functions

gtk_application_new ()

```
GtkApplication *
gtk_application_new (const gchar *application_id,
                     GApplicationFlags flags);
```

Creates a new GtkApplication instance.

When using GtkApplication, it is not necessary to call `gtk_init()` manually. It is called as soon as the application gets registered as the primary instance.

Concretely, `gtk_init()` is called in the default handler for the “startup” signal. Therefore, GtkApplication subclasses should chain up in their “startup” handler before using any GTK+ API.

Note that commandline arguments are not passed to `gtk_init()`. All GTK+ functionality that is available via commandline arguments can also be achieved by setting suitable environment variables such as `G_DEBUG`, so this should not be a big problem. If you absolutely must support GTK+ commandline arguments, you can explicitly call `gtk_init()` before creating the application instance.

If non-`NULL`, the application ID must be valid. See `g_application_id_is_valid()`.

If no application ID is given then some features (most notably application uniqueness) will be disabled. A null application ID is only allowed with GTK+ 3.6 or later.

Parameters

application_id	The application ID.	[allow-none]
flags	the application flags	

Returns

a new GtkApplication instance

Since: 3.0

---

gtk_application_add_window ()

```
void
gtk_application_add_window (GtkApplication *application,
                            GtkWindow *window);
```

Adds a window to `application` .

This call can only happen after the `application` has started; typically, you should add new application windows in response to the emission of the “activate” signal.

This call is equivalent to setting the “application” property of `window` to `application` .

Normally, the connection between the application and the window will remain until the window is destroyed, but you can explicitly remove it with `gtk_application_remove_window()`.

GTK+ will keep the `application` running as long as it has any windows.

Parameters

application	a GtkApplication
window	a GtkWindow

Since: 3.0

---

gtk_application_remove_window ()

```
void
gtk_application_remove_window (GtkApplication *application,
                               GtkWindow *window);
```

Remove a window from `application` .

If `window` belongs to `application` then this call is equivalent to setting the “application” property of `window` to `NULL`.

The application may stop running as a result of a call to this function.

Parameters

application	a GtkApplication
window	a GtkWindow

Since: 3.0

---

gtk_application_get_windows ()

```
GList *
gtk_application_get_windows (GtkApplication *application);
```

Gets a list of the GtkWindows associated with `application` .

The list is sorted by most recently focused window, such that the first element is the currently focused window. (Useful for choosing a parent for a transient window.)

The list that is returned should not be modified in any way. It will only remain valid until the next focus change or window creation or deletion.

Parameters

application	a GtkApplication

Returns

a GList of GtkWindow.

[element-type GtkWindow][transfer none]

Since: 3.0

---

gtk_application_get_window_by_id ()

```
GtkWindow *
gtk_application_get_window_by_id (GtkApplication *application,
                                  guint id);
```

Returns the GtkApplicationWindow with the given ID.

The ID of a GtkApplicationWindow can be retrieved with `gtk_application_window_get_id()`.

Parameters

application	a GtkApplication
id	an identifier number

Returns

the window with ID `id` , or `NULL` if there is no window with this ID.

[nullable][transfer none]

Since: 3.6

---

gtk_application_get_active_window ()

```
GtkWindow *
gtk_application_get_active_window (GtkApplication *application);
```

Gets the “active” window for the application.

The active window is the one that was most recently focused (within the application). This window may not have the focus at the moment if another application has it — this is just the most recently-focused window within this application.

Parameters

application	a GtkApplication

Returns

the active window, or `NULL` if there isn't one.

[transfer none][nullable]

Since: 3.6

gtk_application_inhibit ()

```
guint
gtk_application_inhibit (GtkApplication *application,
                         GtkWindow *window,
                         GtkApplicationInhibitFlags flags,
                         const gchar *reason);
```

Inform the session manager that certain types of actions should be inhibited. This is not guaranteed to work on all platforms and for all types of actions.

Applications should invoke this method when they begin an operation that should not be interrupted, such as creating a CD or DVD. The types of actions that may be blocked are specified by the `flags` parameter. When the application completes the operation it should call `gtk_application_uninhibit()` to remove the inhibitor. Note that an application can have multiple inhibitors, and all of them must be individually removed. Inhibitors are also cleared when the application exits.

Applications should not expect that they will always be able to block the action. In most cases, users will be given the option to force the action to take place.

Reasons should be short and to the point.

If `window` is given, the session manager may point the user to this window to find out more about why the action is inhibited.

Parameters

application	the GtkApplication	
window	a GtkWindow, or `NULL`.	[allow-none]
flags	what types of actions should be inhibited	
reason	a short, human-readable string that explains why these operations are inhibited.	[allow-none]

Returns

A non-zero cookie that is used to uniquely identify this request. It should be used as an argument to `gtk_application_uninhibit()` in order to remove the request. If the platform does not support inhibiting or the request failed for some reason, 0 is returned.

Since: 3.4

gtk_application_uninhibit ()

```
void
gtk_application_uninhibit (GtkApplication *application,
                           guint cookie);
```

Removes an inhibitor that has been established with `gtk_application_inhibit()`. Inhibitors are also cleared when the application exits.

Parameters

application	the GtkApplication
cookie	a cookie that was returned by `gtk_application_inhibit()`

Since: 3.4

---

gtk_application_is_inhibited ()

```
gboolean
gtk_application_is_inhibited (GtkApplication *application,
                              GtkApplicationInhibitFlags flags);
```

Determines if any of the actions specified in `flags` are currently inhibited (possibly by another application).

Note that this information may not be available (for example when the application is running in a sandbox).

Parameters

application	the GtkApplication
flags	what types of actions should be queried

Returns

`TRUE` if any of the actions specified in `flags` are inhibited

Since: 3.4

---

gtk_application_prefers_app_menu ()

```
gboolean
gtk_application_prefers_app_menu (GtkApplication *application);
```

Determines if the desktop environment in which the application is running would prefer an application menu be shown.

If this function returns `TRUE` then the application should call `gtk_application_set_app_menu()` with the contents of an application menu, which will be shown by the desktop environment. If it returns `FALSE` then you should consider using an alternate approach, such as a menubar.

The value returned by this function is purely advisory and you are free to ignore it. If you call `gtk_application_set_app_menu()` even if the desktop environment doesn't support app menus, then a fallback will be provided.

Applications are similarly free not to set an app menu even if the desktop environment wants to show one. In that case, a fallback will also be created by the desktop environment (GNOME, for example, uses a menu with only a "Quit" item in it).

The value returned by this function never changes. Once it returns a particular value, it is guaranteed to always return the same value.

You may only call this function after the application has been registered and after the base startup handler has run. You're most likely to want to use this from your own startup handler. It may also make sense to consult this function while constructing UI (in activate, open or an action activation handler) in order to determine if you should show a gear menu or not.

This function will return `FALSE` on Mac OS and a default app menu will be created automatically with the "usual" contents of that menu typical to most Mac OS applications. If you call `gtk_application_set_app_menu()` anyway, then this menu will be replaced with your own.

Parameters

application	a GtkApplication

Returns

`TRUE` if you should set an app menu

Since: 3.14

---

gtk_application_get_app_menu ()

```
GMenuModel *
gtk_application_get_app_menu (GtkApplication *application);
```

Returns the menu model that has been set with `gtk_application_set_app_menu()`.

Parameters

application	a GtkApplication

Returns

the application menu of `application` or `NULL` if no application menu has been set.

[transfer none][nullable]

Since: 3.4

---

gtk_application_set_app_menu ()

```
void
gtk_application_set_app_menu (GtkApplication *application,
                              GMenuModel *app_menu);
```

Sets or unsets the application menu for `application` .

This can only be done in the primary instance of the application, after it has been registered. “startup” is a good place to call this.

The application menu is a single menu containing items that typically impact the application as a whole, rather than acting on a specific window or document. For example, you would expect to see “Preferences” or “Quit” in an application menu, but not “Save” or “Print”.

If supported, the application menu will be rendered by the desktop environment.

Use the base GActionMap interface to add actions, to respond to the user selecting these menu items.

Parameters

application	a GtkApplication	
app_menu	a GMenuModel, or `NULL`.	[allow-none]

Since: 3.4

---

gtk_application_get_menubar ()

```
GMenuModel *
gtk_application_get_menubar (GtkApplication *application);
```

Returns the menu model that has been set with `gtk_application_set_menubar()`.

Parameters

application	a GtkApplication

Returns

the menubar for windows of `application` .

[transfer none]

Since: 3.4

---

gtk_application_set_menubar ()

```
void
gtk_application_set_menubar (GtkApplication *application,
                             GMenuModel *menubar);
```

Sets or unsets the menubar for windows of `application`.

This is a menubar in the traditional sense.

This can only be done in the primary instance of the application, after it has been registered. “startup” is a good place to call this.

Depending on the desktop environment, this may appear at the top of each window, or at the top of the screen. In some environments, if both the application menu and the menubar are set, the application menu will be presented as if it were the first item of the menubar. Other environments treat the two as completely separate — for example, the application menu may be rendered by the desktop shell while the menubar (if set) remains in each individual window.

Use the base GActionMap interface to add actions, to respond to the user selecting these menu items.

Parameters

application	a GtkApplication	
menubar	a GMenuModel, or `NULL`.	[allow-none]

Since: 3.4

---

gtk_application_get_menu_by_id ()

```
GMenu *
gtk_application_get_menu_by_id (GtkApplication *application,
                                const gchar *id);
```

Gets a menu from automatically loaded resources. See Automatic resources for more information.

Parameters

application	a GtkApplication
id	the id of the menu to look up

Returns

Gets the menu with the given id from the automatically loaded resources.

[transfer none]

Since: 3.14

---

gtk_application_add_accelerator ()

```
void
gtk_application_add_accelerator (GtkApplication *application,
                                 const gchar *accelerator,
                                 const gchar *action_name,
                                 GVariant *parameter);
```

`gtk_application_add_accelerator` has been deprecated since version 3.14 and should not be used in newly-written code.

Use `gtk_application_set_accels_for_action()` instead

Installs an accelerator that will cause the named action to be activated when the key combination specificed by `accelerator` is pressed.

`accelerator` must be a string that can be parsed by `gtk_accelerator_parse()`, e.g. "<Primary>q" or “<Control><Alt>p”.

`action_name` must be the name of an action as it would be used in the app menu, i.e. actions that have been added to the application are referred to with an “app.” prefix, and window-specific actions with a “win.” prefix.

GtkApplication also extracts accelerators out of “accel” attributes in the GMenuModels passed to `gtk_application_set_app_menu()` and `gtk_application_set_menubar()`, which is usually more convenient than calling this function for each accelerator.

Parameters

application	a GtkApplication	
accelerator	accelerator string	
action_name	the name of the action to activate	
parameter	parameter to pass when activating the action, or `NULL` if the action does not accept an activation parameter.	[allow-none]

Since: 3.4

---

gtk_application_remove_accelerator ()

```
void
gtk_application_remove_accelerator (GtkApplication *application,
                                    const gchar *action_name,
                                    GVariant *parameter);
```

`gtk_application_remove_accelerator` has been deprecated since version 3.14 and should not be used in newly-written code.

Use `gtk_application_set_accels_for_action()` instead

Removes an accelerator that has been previously added with `gtk_application_add_accelerator()`.

Parameters

application	a GtkApplication	
action_name	the name of the action to activate	
parameter	parameter to pass when activating the action, or `NULL` if the action does not accept an activation parameter.	[allow-none]

Since: 3.4

---

gtk_application_list_action_descriptions ()

```
gchar **
gtk_application_list_action_descriptions
                                (GtkApplication *application);
```

Lists the detailed action names which have associated accelerators. See `gtk_application_set_accels_for_action()`.

Parameters

application	a GtkApplication

Returns

a `NULL`-terminated array of strings, free with `g_strfreev()` when done.

[transfer full]

Since: 3.12

---

gtk_application_get_accels_for_action ()

```
gchar **
gtk_application_get_accels_for_action (GtkApplication *application,
                                       const gchar *detailed_action_name);
```

Gets the accelerators that are currently associated with the given action.

Parameters

application	a GtkApplication
detailed_action_name	a detailed action name, specifying an action and target to obtain accelerators for

Returns

accelerators for `detailed_action_name` , as a `NULL`-terminated array. Free with `g_strfreev()` when no longer needed.

[transfer full]

Since: 3.12

---

gtk_application_set_accels_for_action ()

```
void
gtk_application_set_accels_for_action (GtkApplication *application,
                                       const gchar *detailed_action_name,
                                       const gchar * const *accels);
```

Sets zero or more keyboard accelerators that will trigger the given action. The first item in `accels` will be the primary accelerator, which may be displayed in the UI.

To remove all accelerators for an action, use an empty, zero-terminated array for `accels` .

For the `detailed_action_name` , see `g_action_parse_detailed_name()` and `g_action_print_detailed_name()`.

Parameters

application	a GtkApplication	
detailed_action_name	a detailed action name, specifying an action and target to associate accelerators with	
accels	a list of accelerators in the format understood by `gtk_accelerator_parse()`.	[array zero-terminated=1]

Since: 3.12

gtk_application_get_actions_for_accel ()

```
gchar **
gtk_application_get_actions_for_accel (GtkApplication *application,
                                       const gchar *accel);
```

Returns the list of actions (possibly empty) that `accel` maps to. Each item in the list is a detailed action name in the usual form.

This might be useful to discover if an accel already exists in order to prevent installation of a conflicting accelerator (from an accelerator editor or a plugin system, for example). Note that having more than one action per accelerator may not be a bad thing and might make sense in cases where the actions never appear in the same context.

In case there are no actions for a given accelerator, an empty array is returned. `NULL` is never returned.

It is a programmer error to pass an invalid accelerator string. If you are unsure, check it with `gtk_accelerator_parse()` first.

Parameters

application	a GtkApplication
accel	an accelerator that can be parsed by `gtk_accelerator_parse()`

Returns

a `NULL`-terminated array of actions for `accel`.

[transfer full]

Since: 3.14

Types and Values

struct GtkApplication

```
struct GtkApplication;
```

---

struct GtkApplicationClass

```
struct GtkApplicationClass {
  GApplicationClass parent_class;


  void (*window_added)   (GtkApplication *application,
                          GtkWindow      *window);
  void (*window_removed) (GtkApplication *application,
                          GtkWindow      *window);
};
```

Members

`window_added` ()	Signal emitted when a GtkWindow is added to application through `gtk_application_add_window()`.
`window_removed` ()	Signal emitted when a GtkWindow is removed from application, either as a side-effect of being destroyed or explicitly through `gtk_application_remove_window()`.

---

enum GtkApplicationInhibitFlags

Types of user actions that may be blocked by `gtk_application_inhibit()`.

Members

GTK_APPLICATION_INHIBIT_LOGOUT	Inhibit ending the user session by logging out or by shutting down the computer
GTK_APPLICATION_INHIBIT_SWITCH	Inhibit user switching
GTK_APPLICATION_INHIBIT_SUSPEND	Inhibit suspending the session or computer
GTK_APPLICATION_INHIBIT_IDLE	Inhibit the session being marked as idle (and possibly locked)

Since: 3.4

Property Details

The “active-window” property

“active-window” GtkWindow *

The window which most recently had focus.

Flags: Read

The “app-menu” property

“app-menu” GMenuModel *

The GMenuModel for the application menu.

Flags: Read / Write

The “menubar” property

“menubar” GMenuModel *

The GMenuModel for the menubar.

Flags: Read / Write

The “register-session” property

“register-session” gboolean

Set this property to TRUE to register with the session manager.

Flags: Read / Write

Default value: FALSE

Since: 3.4

The “screensaver-active” property

“screensaver-active” gboolean

This property is TRUE if GTK+ believes that the screensaver is currently active. GTK+ only tracks session state (including this) when “register-session” is set to TRUE.

Tracking the screensaver state is supported on Linux.

Flags: Read

Default value: FALSE

Since: 3.24

Signal Details

The “query-end” signal

```
void
user_function (GtkApplication *application,
               gpointer        user_data)
```

Emitted when the session manager is about to end the session, only if “register-session” is TRUE. Applications can connect to this signal and call gtk_application_inhibit() with GTK_APPLICATION_INHIBIT_LOGOUT to delay the end of the session until state has been saved.

Parameters

application	the GtkApplication which emitted the signal
user_data	user data set when the signal handler was connected.

Flags: Run First

Since: 3.24.8

The “window-added” signal

```
void
user_function (GtkApplication *application,
               GtkWindow      *window,
               gpointer        user_data)
```

Emitted when a GtkWindow is added to application through gtk_application_add_window().

Parameters

application	the GtkApplication which emitted the signal
window	the newly-added GtkWindow
user_data	user data set when the signal handler was connected.

Flags: Run First

Since: 3.2

The “window-removed” signal

```
void
user_function (GtkApplication *application,
               GtkWindow      *window,
               gpointer        user_data)
```

Emitted when a GtkWindow is removed from `application` , either as a side-effect of being destroyed or explicitly through `gtk_application_remove_window()`.

Parameters

application	the GtkApplication which emitted the signal
window	the GtkWindow that is being removed
user_data	user data set when the signal handler was connected.

Flags: Run First

Since: 3.2

GtkApplicationWindow

GtkApplicationWindow — GtkWindow subclass with GtkApplication support

Functions

GtkWidget *	gtk_application_window_new ()
void	gtk_application_window_set_show_menubar ()
gboolean	gtk_application_window_get_show_menubar ()
guint	gtk_application_window_get_id ()
void	gtk_application_window_set_help_overlay ()
GtkShortcutsWindow *	gtk_application_window_get_help_overlay ()

Properties

gboolean	show-menubar	Read / Write / Construct

Types and Values

struct	GtkApplicationWindow
struct	GtkApplicationWindowClass

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkContainer
            └── GtkBin
                └── GtkWindow
                    └── GtkApplicationWindow
```

Implemented Interfaces

GtkApplicationWindow implements AtkImplementorIface, GtkBuildable, GActionGroup and GActionMap.

Includes

```
#include <gtk/gtk.h>
```

Description

GtkApplicationWindow is a GtkWindow subclass that offers some extra functionality for better integration with GtkApplication features. Notably, it can handle both the application menu as well as the menubar. See `gtk_application_set_app_menu()` and `gtk_application_set_menubar()`.

This class implements the GActionGroup and GActionMap interfaces, to let you add window-specific actions that will be exported by the associated GtkApplication, together with its application-wide actions. Window-specific actions are prefixed with the “win.” prefix and application-wide actions are prefixed with the “app.” prefix. Actions must be addressed with the prefixed name when referring to them from a GMenuModel.

Note that widgets that are placed inside a GtkApplicationWindow can also activate these actions, if they implement the GtkActionable interface.

As with GtkApplication, the GDK lock will be acquired when processing actions arriving from other processes and should therefore be held when activating actions locally (if GDK threads are enabled).

The settings “gtk-shell-shows-app-menu” and “gtk-shell-shows-menubar” tell GTK+ whether the desktop environment is showing the application menu and menubar models outside the application as part of the desktop shell. For instance, on OS X, both menus will be displayed remotely; on Windows neither will be. gnome-shell (starting with version 3.4) will display the application menu, but not the menubar.

If the desktop environment does not display the menubar, then GtkApplicationWindow will automatically show a GtkMenuBar for it. This behaviour can be overridden with the “show-menubar” property. If the desktop environment does not display the application menu, then it will automatically be included in the menubar or in the windows client-side decorations.

A GtkApplicationWindow with a menubar

```
1   GtkApplication *app = gtk_application_new ("org.gtk.test", 0);
2
3   GtkBuilder *builder = gtk_builder_new_from_string (
4       "<interface>"
5       "  <menu id='menubar'>"
6       "    <submenu label='_Edit'>"
7       "      <item label='_Copy' action='win.copy'/>"
8       "      <item label='_Paste' action='win.paste'/>"
9       "    </submenu>"
10      "  </menu>"
11      "</interface>",
12      -1);
13
14  GMenuModel *menubar = G_MENU_MODEL (gtk_builder_get_object (builder,
15                                                              "menubar"));
16  gtk_application_set_menubar (GTK_APPLICATION (app), menubar);
17  g_object_unref (builder);
18
19  // ...
20
21  GtkWidget *window = gtk_application_window_new (app);
```

Handling fallback yourself

A simple example

The XML format understood by GtkBuilder for GMenuModel consists of a toplevel `<menu>` element, which contains one or more `<item>` elements. Each `<item>` element contains `<attribute>` and `<link>` elements with a mandatory name attribute. `<link>` elements have the same content model as `<menu>`. Instead of `<link name="submenu>` or `<link name="section">`, you can use `<submenu>` or `<section>` elements.

Attribute values can be translated using gettext, like other GtkBuilder content. `<attribute>` elements can be marked for translation with a `translatable="yes"` attribute. It is also possible to specify message context and translator comments, using the context and comments attributes. To make use of this, the GtkBuilder must have been given the gettext domain to use.

The following attributes are used when constructing menu items:

- "label": a user-visible string to display
- "action": the prefixed name of the action to trigger
- "target": the parameter to use when activating the action
- "icon" and "verb-icon": names of icons that may be displayed
- "submenu-action": name of an action that may be used to determine if a submenu can be opened
- "hidden-when": a string used to determine when the item will be hidden. Possible values include "action-disabled", "action-missing", "macos-menubar".

The following attributes are used when constructing sections:

- "label": a user-visible string to use as section heading
- "display-hint": a string used to determine special formatting for the section. Possible values include "horizontal-buttons".
- "text-direction": a string used to determine the GtkTextDirection to use when "display-hint" is set to "horizontal-buttons". Possible values include "rtl", "ltr", and "none".

The following attributes are used when constructing submenus:

- "label": a user-visible string to display
- "icon": icon name to display

Functions

gtk_application_window_new ()

```
GtkWidget *
gtk_application_window_new (GtkApplication *application);
```

Creates a new GtkApplicationWindow.

Parameters

application	a GtkApplication

Returns

a newly created GtkApplicationWindow

Since: 3.4

---

gtk_application_window_set_show_menubar ()

```
void
gtk_application_window_set_show_menubar
                                (GtkApplicationWindow *window,
                                 gboolean show_menubar);
```

Sets whether the window will display a menubar for the app menu and menubar as needed.

Parameters

window	a GtkApplicationWindow
show_menubar	whether to show a menubar when needed

Since: 3.4

---

gtk_application_window_get_show_menubar ()

```
gboolean
gtk_application_window_get_show_menubar
                                (GtkApplicationWindow *window);
```

Returns whether the window will display a menubar for the app menu and menubar as needed.

Parameters

window	a GtkApplicationWindow

Returns

`TRUE` if `window` will display a menubar when needed

Since: 3.4

---

gtk_application_window_get_id ()

```
guint
gtk_application_window_get_id (GtkApplicationWindow *window);
```

Returns the unique ID of the window. If the window has not yet been added to a GtkApplication, returns `0`.

Parameters

window	a GtkApplicationWindow

Returns

the unique ID for `window` , or `0` if the window has not yet been added to a GtkApplication

Since: 3.6

---

gtk_application_window_set_help_overlay ()

```
void
gtk_application_window_set_help_overlay
                                (GtkApplicationWindow *window,
                                 GtkShortcutsWindow *help_overlay);
```

Associates a shortcuts window with the application window, and sets up an action with the name win.show-help-overlay to present it.

`window` takes resposibility for destroying `help_overlay` .

Parameters

window	a GtkApplicationWindow	
help_overlay	a GtkShortcutsWindow.	[nullable]

Since: 3.20

---

gtk_application_window_get_help_overlay ()

```
GtkShortcutsWindow *
gtk_application_window_get_help_overlay
                                (GtkApplicationWindow *window);
```

Gets the GtkShortcutsWindow that has been set up with a prior call to `gtk_application_window_set_help_overlay()`.

Parameters

window	a GtkApplicationWindow

Returns

the help overlay associated with `window` , or `NULL`.

[transfer none][nullable]

Since: 3.20

Types and Values

struct GtkApplicationWindow

```
struct GtkApplicationWindow;
```

---

struct GtkApplicationWindowClass

```
struct GtkApplicationWindowClass {
  GtkWindowClass parent_class;
};
```

Members

Property Details

The “show-menubar” property

```
  “show-menubar”             gboolean
```

If this property is TRUE, the window will display a menubar that includes the app menu and menubar, unless these are shown by the desktop shell. See gtk_application_set_app_menu() and gtk_application_set_menubar().

If FALSE, the window will not display a menubar, regardless of whether the desktop shell is showing the menus or not.

Flags: Read / Write / Construct

Default value: TRUE

GtkActionable

GtkActionable — An interface for widgets that can be associated with actions

Functions

const gchar *	gtk_actionable_get_action_name ()
void	gtk_actionable_set_action_name ()
GVariant *	gtk_actionable_get_action_target_value ()
void	gtk_actionable_set_action_target_value ()
void	gtk_actionable_set_action_target ()
void	gtk_actionable_set_detailed_action_name ()

Properties

gchar *	action-name	Read / Write
GVariant *	action-target	Read / Write

Types and Values

	GtkActionable
struct	GtkActionableInterface

Object Hierarchy

```
GInterface
└── GtkActionable
```

Prerequisites

GtkActionable requires GtkWidget.

Known Implementations

GtkActionable is implemented by GtkButton, GtkCheckButton, GtkCheckMenuItem, GtkColorButton, GtkFontButton, GtkImageMenuItem, GtkLinkButton, GtkListBoxRow, GtkLockButton, GtkMenuButton, GtkMenuItem, GtkMenuToolButton, GtkModelButton, GtkRadioButton, GtkRadioMenuItem, GtkRadioToolButton, GtkScaleButton, GtkSeparatorMenuItem, GtkSwitch, GtkTearoffMenuItem, GtkToggleButton, GtkToggleToolButton, GtkToolButton and GtkVolumeButton.

Includes

```
#include <gtk/gtk.h>
```

Description

This interface provides a convenient way of associating widgets with actions on a GtkApplicationWindow or GtkApplication.

It primarily consists of two properties: “action-name” and “action-target”. There are also some convenience APIs for setting these properties.

The action will be looked up in action groups that are found among the widgets ancestors. Most commonly, these will be the actions with the “win.” or “app.” prefix that are associated with the GtkApplicationWindow or GtkApplication, but other action groups that are added with `gtk_widget_insert_action_group()` will be consulted as well.

Functions

gtk_actionable_get_action_name ()

```
const gchar *
gtk_actionable_get_action_name (GtkActionable *actionable);
```

Gets the action name for `actionable` .

See `gtk_actionable_set_action_name()` for more information.

Parameters

actionable	a GtkActionable widget

Returns

the action name, or `NULL` if none is set.

[nullable]

Since: 3.4

gtk_actionable_set_action_name ()

```
void
gtk_actionable_set_action_name (GtkActionable *actionable,
                                const gchar *action_name);
```

Specifies the name of the action with which this widget should be associated. If `action_name` is `NULL` then the widget will be unassociated from any previous action.

Usually this function is used when the widget is located (or will be located) within the hierarchy of a GtkApplicationWindow.

Names are of the form “win.save” or “app.quit” for actions on the containing GtkApplicationWindow or its associated GtkApplication, respectively. This is the same form used for actions in the GMenu associated with the window.

Parameters

actionable	a GtkActionable widget	
action_name	an action name, or `NULL`.	[nullable]

Since: 3.4

---

gtk_actionable_get_action_target_value ()

```
GVariant *
gtk_actionable_get_action_target_value
                               (GtkActionable *actionable);
```

Gets the current target value of `actionable` .

See `gtk_actionable_set_action_target_value()` for more information.

Parameters

actionable	a GtkActionable widget

Returns

the current target value.

[transfer none]

Since: 3.4

---

gtk_actionable_set_action_target_value ()

```
void
gtk_actionable_set_action_target_value
                                (GtkActionable *actionable,
                                 GVariant *target_value);
```

Sets the target value of an actionable widget.

If `target_value` is `NULL` then the target value is unset.

The target value has two purposes. First, it is used as the parameter to activation of the action associated with the GtkActionable widget. Second, it is used to determine if the widget should be rendered as “active” — the widget is active if the state is equal to the given target.

Consider the example of associating a set of buttons with a GAction with string state in a typical “radio button” situation. Each button will be associated with the same action, but with a different target value for that action. Clicking on a particular button will activate the action with the target of that button, which will typically cause the action’s state to change to that value. Since the action’s state is now equal to the target value of the button, the button will now be rendered as active (and the other buttons, with different targets, rendered inactive).

Parameters

actionable	a GtkActionable widget
target_value	a GVariant to set as the target value, [nullable] or `NULL`.

Since: 3.4

---

gtk_actionable_set_action_target ()

```
void
gtk_actionable_set_action_target (GtkActionable *actionable,
                                  const gchar *format_string,
                                  ...);
```

Sets the target of an actionable widget.

This is a convenience function that calls `g_variant_new()` for `format_string` and uses the result to call `gtk_actionable_set_action_target_value()`.

If you are setting a string-valued target and want to set the action name at the same time, you can use `gtk_actionable_set_detailed_action_name()`.

Parameters

actionable	a GtkActionable widget
format_string	a GVariant format string
...	arguments appropriate for `format_string`

Since: 3.4

---

gtk_actionable_set_detailed_action_name ()

```
void
gtk_actionable_set_detailed_action_name
                                (GtkActionable *actionable,
                                 const gchar *detailed_action_name);
```

Sets the action-name and associated string target value of an actionable widget.

`detailed_action_name` is a string in the format accepted by `g_action_parse_detailed_name()`.

(Note that prior to version 3.22.25, this function is only usable for actions with a simple "s" target, and `detailed_action_name` must be of the form `"action::target"` where `action` is the action name and `target` is the string to use as the target.)

Parameters

actionable	a GtkActionable widget
detailed_action_name	the detailed action name

Since: 3.4

Types and Values

GtkActionable

```
typedef struct _GtkActionable GtkActionable;
```

An opaque pointer type.

---

struct GtkActionableInterface

```
struct GtkActionableInterface {
  const gchar * (* get_action_name)            (GtkActionable *actionable);
  void          (* set_action_name)            (GtkActionable *actionable,
                                                const gchar   *action_name);
  GVariant *    (* get_action_target_value)    (GtkActionable *actionable);
  void          (* set_action_target_value)    (GtkActionable *actionable,
                                                GVariant      *target_value);
};
```

The interface vtable for GtkActionable.

Members

`get_action_name` ()	virtual function for `gtk_actionable_get_action_name()`
`set_action_name` ()	virtual function for `gtk_actionable_set_action_name()`
`get_action_target_value` ()	virtual function for `gtk_actionable_get_action_target_value()`
`set_action_target_value` ()	virtual function for `gtk_actionable_set_action_target_value()`

Property Details

The `“action-name”` property

```
  “action-name”              gchar *
```

The name of the associated action, like 'app.quit'.

Flags: Read / Write

Default value: NULL

The `“action-target”` property

```
  “action-target”            GVariant *
```

The parameter for action invocations.

Flags: Read / Write

Allowed values: GVariant<*>

Default value: NULL

Interface builder

GtkBuilder — Build an interface from an XML UI definition

GtkBuildable — Interface for objects that can be built by GtkBuilder

---

GtkBuilder

GtkBuilder — Build an interface from an XML UI definition

Functions

void	(*GtkBuilderConnectFunc) ()
GtkBuilder *	gtk_builder_new ()
GtkBuilder *	gtk_builder_new_from_file ()
GtkBuilder *	gtk_builder_new_from_resource ()
GtkBuilder *	gtk_builder_new_from_string ()
void	gtk_builder_add_callback_symbol ()
void	gtk_builder_add_callback_symbols ()
GCallback	gtk_builder_lookup_callback_symbol ()
guint	gtk_builder_add_from_file ()
guint	gtk_builder_add_from_resource ()
guint	gtk_builder_add_from_string ()
guint	gtk_builder_add_objects_from_file ()
guint	gtk_builder_add_objects_from_string ()
guint	gtk_builder_add_objects_from_resource ()
guint	gtk_builder_extend_with_template ()
GObject *	gtk_builder_get_object ()
GSList *	gtk_builder_get_objects ()
void	gtk_builder_expose_object ()
void	gtk_builder_connect_signals ()
void	gtk_builder_connect_signals_full ()
void	gtk_builder_set_translation_domain ()
const gchar *	gtk_builder_get_translation_domain ()
void	gtk_builder_set_application ()
GtkApplication *	gtk_builder_get_application ()
GType	gtk_builder_get_type_from_name ()
gboolean	gtk_builder_value_from_string ()
gboolean	gtk_builder_value_from_string_type ()
#define	GTK_BUILDER_WARN_INVALID_CHILD_TYPE()

Properties

gchar *	translation-domain	Read / Write

Types and Values

	GtkBuilder
enum	GtkBuilderError
#define	GTK_BUILDER_ERROR

Object Hierarchy

```
GObject
└── GtkBuilder
```

Includes

```
#include <gtk/gtk.h>
```

Description

A GtkBuilder is an auxiliary object that reads textual descriptions of a user interface and instantiates the described objects. To create a GtkBuilder from a user interface description, call `gtk_builder_new_from_file()`, `gtk_builder_new_from_resource()` or `gtk_builder_new_from_string()`.

In the (unusual) case that you want to add user interface descriptions from multiple sources to the same GtkBuilder you can call `gtk_builder_new()` to get an empty builder and populate it by (multiple) calls to `gtk_builder_add_from_file()`, `gtk_builder_add_from_resource()` or `gtk_builder_add_from_string()`.

A GtkBuilder holds a reference to all objects that it has constructed and drops these references when it is finalized. This finalization can cause the destruction of non-widget objects or widgets which are not contained in a toplevel window. For toplevel windows constructed by a builder, it is the responsibility of the user to call `gtk_widget_destroy()` to get rid of them and all the widgets they contain.

The functions `gtk_builder_get_object()` and `gtk_builder_get_objects()` can be used to access the widgets in the interface by the names assigned to them inside the UI description. Toplevel windows returned by these functions will stay around until the user explicitly destroys them with `gtk_widget_destroy()`. Other widgets will either be part of a larger hierarchy constructed by the builder (in which case you should not have to worry about their lifecycle), or without a parent, in which case they have to be added to some container to make use of them. Non-widget objects need to be reffed with `g_object_ref()` to keep them beyond the lifespan of the builder.

The function `gtk_builder_connect_signals()` and variants thereof can be used to connect handlers to the named signals in the description.

GtkBuilder UI Definitions

GtkBuilder parses textual descriptions of user interfaces which are specified in an XML format which can be roughly described by the RELAX NG schema below. We refer to these descriptions as “GtkBuilder UI definitions” or just “UI definitions” if the context is clear. Do not confuse GtkBuilder UI Definitions with GtkUIManager UI Definitions, which are more limited in scope. It is common to use `.ui` as the filename extension for files containing GtkBuilder UI definitions.

RELAX NG Compact Syntax

The toplevel element is <interface>. It optionally takes a “domain” attribute, which will make the builder look for translated strings using `dgettext()` in the domain specified. This can also be done by calling `gtk_builder_set_translation_domain()` on the builder. Objects are described by <object> elements, which can contain <property> elements to set properties, <signal> elements which connect signals to handlers, and <child> elements, which describe child objects (most often widgets inside a container, but also e.g. actions in an action group, or columns in a tree model). A <child> element contains an <object> element which describes the child object. The target toolkit version(s) are described by <requires> elements, the “lib” attribute specifies the widget library in question (currently the only supported value is “gtk+”) and the “version” attribute specifies the target version in the form “<major>.<minor>”. The builder will error out if the version requirements are not met.

Typically, the specific kind of object represented by an <object> element is specified by the “class” attribute. If the type has not been loaded yet, GTK+ tries to find the `get_type()` function from the class name by applying heuristics. This works in most cases, but if necessary, it is possible to specify the name of the `get_type()` function explictly with the "type-func" attribute. As a special case, GtkBuilder allows to use an object that has been constructed by a GtkUIManager in another part of the UI definition by specifying the id of the GtkUIManager in the “constructor” attribute and the name of the object in the “id” attribute.

Objects may be given a name with the “id” attribute, which allows the application to retrieve them from the builder with `gtk_builder_get_object()`. An id is also necessary to use the object as property value in other parts of the UI definition. GTK+ reserves ids starting and ending with ___ (3 underscores) for its own purposes.

Setting properties of objects is pretty straightforward with the <property> element: the “name” attribute specifies the name of the property, and the content of the element specifies the value. If the “translatable” attribute is set to a true value, GTK+ uses `gettext()` (or `dgettext()` if the builder has a translation domain set) to find a translation for the value. This happens before the value is parsed, so it can be used for properties of any type, but it is probably most useful for string properties. It is also possible to specify a context to disambiguate short strings, and comments which may help the translators.

GtkBuilder can parse textual representations for the most common property types: characters, strings, integers, floating-point numbers, booleans (strings like “TRUE”, “t”, “yes”, “y”, “1” are interpreted as `TRUE`, strings like “FALSE”, “f”, “no”, “n”, “0” are interpreted as `FALSE`), enumerations (can be specified by their name, nick or integer value), flags (can be specified by their name, nick, integer value, optionally combined with “|”, e.g. “GTK_VISIBLE|GTK_REALIZED”) and colors (in a format understood by `gdk_rgba_parse()`).

GVariants can be specified in the format understood by `g_variant_parse()`, and pixbufs can be specified as a filename of an image file to load.

Objects can be referred to by their name and by default refer to objects declared in the local xml fragment and objects exposed via `gtk_builder_expose_object()`. In general, GtkBuilder allows forward references to objects — declared in the local xml; an object doesn’t have to be constructed before it can be referred to. The exception to this rule is that an object has to be constructed before it can be used as the value of a construct-only property.

It is also possible to bind a property value to another object's property value using the attributes "bind-source" to specify the source object of the binding, "bind-property" to specify the source property and optionally "bind-flags" to specify the binding flags Internally builder implement this using GBinding objects. For more information see `g_object_bind_property()`

Signal handlers are set up with the <signal> element. The “name” attribute specifies the name of the signal, and the “handler” attribute specifies the function to connect to the signal. By default, GTK+ tries to find the handler using `g_module_symbol()`, but this can be changed by passing a custom GtkBuilderConnectFunc to `gtk_builder_connect_signals_full()`. The remaining attributes, “after”, “swapped” and “object”, have the same meaning as the corresponding parameters of the `g_signal_connect_object()` or `g_signal_connect_data()` functions. A “last_modification_time” attribute is also allowed, but it does not have a meaning to the builder.

Sometimes it is necessary to refer to widgets which have implicitly been constructed by GTK+ as part of a composite widget, to set properties on them or to add further children (e.g. the `vbox` of a GtkDialog). This can be achieved by setting the “internal-child” propery of the <child> element to a true value. Note that GtkBuilder still requires an <object> element for the internal child, even if it has already been constructed.

A number of widgets have different places where a child can be added (e.g. tabs vs. page content in notebooks). This can be reflected in a UI definition by specifying the “type” attribute on a <child> The possible values for the “type” attribute are described in the sections describing the widget-specific portions of UI definitions.

A GtkBuilder UI Definition

```
1   <interface>
2     <object class="GtkDialog" id="dialog1">
3       <child internal-child="vbox">
4         <object class="GtkBox" id="vbox1">
5           <property name="border-width">10</property>
6           <child internal-child="action_area">
7             <object class="GtkButtonBox" id="hbuttonbox1">
8               <property name="border-width">20</property>
9               <child>
10                <object class="GtkButton" id="ok_button">
11                  <property name="label">gtk-ok</property>
12                  <property name="use-stock">TRUE</property>
13                  <signal name="clicked" handler="ok_button_clicked"/>
14                </object>
15              </child>
16            </object>
17          </child>
18        </object>
19      </child>
20    </object>
21  </interface>
```

Beyond this general structure, several object classes define their own XML DTD fragments for filling in the ANY placeholders in the DTD above. Note that a custom element in a <child> element gets parsed by the custom tag handler of the parent object, while a custom element in an <object> element gets parsed by the custom tag handler of the object.

These XML fragments are explained in the documentation of the respective objects.

Additionally, since 3.10 a special <template> tag has been added to the format allowing one to define a widget class’s components. See the GtkWidget documentation for details.

Functions

GtkBuilderConnectFunc ()

```
void
(*GtkBuilderConnectFunc) (GtkBuilder *builder,
                          GObject *object,
                          const gchar *signal_name,
                          const gchar *handler_name,
                          GObject *connect_object,
                          GConnectFlags flags,
                          gpointer user_data);
```

This is the signature of a function used to connect signals. It is used by the `gtk_builder_connect_signals()` and `gtk_builder_connect_signals_full()` methods. It is mainly intended for interpreted language bindings, but could be useful where the programmer wants more control over the signal connection process. Note that this function can only be called once, subsequent calls will do nothing.

Parameters

builder	a GtkBuilder	
object	object to connect a signal to	
signal_name	name of the signal	
handler_name	name of the handler	
connect_object	a GObject, if non-`NULL`, use `g_signal_connect_object()`.	[nullable]
flags	GConnectFlags to use	
user_data	user data	

Since: 2.12

---

gtk_builder_new ()

```
GtkBuilder *
gtk_builder_new (void);
```

Creates a new empty builder object.

This function is only useful if you intend to make multiple calls to `gtk_builder_add_from_file()`, `gtk_builder_add_from_resource()` or `gtk_builder_add_from_string()` in order to merge multiple UI descriptions into a single builder.

Most users will probably want to use `gtk_builder_new_from_file()`, `gtk_builder_new_from_resource()` or `gtk_builder_new_from_string()`.

Returns

a new (empty) GtkBuilder object

Since: 2.12

---

gtk_builder_new_from_file ()

```
GtkBuilder *
gtk_builder_new_from_file (const gchar *filename);
```

Builds the GtkBuilder UI definition in the file `filename`.

If there is an error opening the file or parsing the description then the program will be aborted. You should only ever attempt to parse user interface descriptions that are shipped as part of your program.

Parameters

filename	filename of user interface description file

Returns

a GtkBuilder containing the described interface

Since: 3.10

---

gtk_builder_new_from_resource ()

```
GtkBuilder *
gtk_builder_new_from_resource (const gchar *resource_path);
```

Builds the GtkBuilder UI definition at `resource_path`.

If there is an error locating the resource or parsing the description, then the program will be aborted.

Parameters

resource_path	a GResource resource path

Returns

a GtkBuilder containing the described interface

Since: 3.10

---

gtk_builder_new_from_string ()

```
GtkBuilder *
gtk_builder_new_from_string (const gchar *string,
                             gssize length);
```

Builds the user interface described by `string` (in the GtkBuilder UI definition format).

If `string` is `NULL`-terminated, then `length` should be -1. If `length` is not -1, then it is the length of `string` .

If there is an error parsing `string` then the program will be aborted. You should not attempt to parse user interface description from untrusted sources.

Parameters

string	a user interface (XML) description
length	the length of `string` , or -1

Returns

a GtkBuilder containing the interface described by `string`

Since: 3.10

---

gtk_builder_add_callback_symbol ()

```
void
gtk_builder_add_callback_symbol (GtkBuilder *builder,
                                 const gchar *callback_name,
                                 GCallback callback_symbol);
```

Adds the `callback_symbol` to the scope of `builder` under the given `callback_name` .

Using this function overrides the behavior of `gtk_builder_connect_signals()` for any callback symbols that are added. Using this method allows for better encapsulation as it does not require that callback symbols be declared in the global namespace.

Parameters

builder	a GtkBuilder	
callback_name	The name of the callback, as expected in the XML	
callback_symbol	The callback pointer.	[scope async]

Since: 3.10

---

gtk_builder_add_callback_symbols ()

```
void
gtk_builder_add_callback_symbols (GtkBuilder *builder,
                                  const gchar *first_callback_name,
                                  GCallback first_callback_symbol,
                                  ...);
```

A convenience function to add many callbacks instead of calling `gtk_builder_add_callback_symbol()` for each symbol.

Parameters

builder	a GtkBuilder	
first_callback_name	The name of the callback, as expected in the XML	
first_callback_symbol	The callback pointer.	[scope async]
...	A list of callback name and callback symbol pairs terminated with `NULL`	

Since: 3.10

---

gtk_builder_lookup_callback_symbol ()

```
GCallback
gtk_builder_lookup_callback_symbol (GtkBuilder *builder,
                                    const gchar *callback_name);
```

Fetches a symbol previously added to `builder` with `gtk_builder_add_callback_symbols()`

This function is intended for possible use in language bindings or for any case that one might be cusomizing signal connections using `gtk_builder_connect_signals_full()`

[skip]

Parameters

builder	a GtkBuilder
callback_name	The name of the callback

Returns

The callback symbol in `builder` for `callback_name` , or `NULL`.

[nullable]

Since: 3.10

---

gtk_builder_add_from_file ()

```
guint
gtk_builder_add_from_file (GtkBuilder *builder,
                           const gchar *filename,
                           GError **error);
```

Parses a file containing a GtkBuilder UI definition and merges it with the current contents of `builder` .

Most users will probably want to use `gtk_builder_new_from_file()`.

If an error occurs, 0 will be returned and `error` will be assigned a GError from the GTK_BUILDER_ERROR, G_MARKUP_ERROR or G_FILE_ERROR domain.

It's not really reasonable to attempt to handle failures of this call. You should not use this function with untrusted files (ie: files that are not part of your application). Broken GtkBuilder files can easily crash your program, and it's possible that memory was leaked leading up to the reported failure. The only reasonable thing to do when an error is detected is to call `g_error()`.

Parameters

builder	a GtkBuilder
filename	the name of the file to parse
error	return location for an error, or `NULL`. [allow-none]

Returns

A positive value on success, 0 if an error occurred

Since: 2.12

gtk_builder_add_from_resource ()

```
guint
gtk_builder_add_from_resource (GtkBuilder *builder,
                               const gchar *resource_path,
                               GError **error);
```

Parses a resource file containing a GtkBuilder UI definition and merges it with the current contents of `builder` .

Most users will probably want to use `gtk_builder_new_from_resource()`.

If an error occurs, 0 will be returned and `error` will be assigned a GError from the GTK_BUILDER_ERROR, G_MARKUP_ERROR or G_RESOURCE_ERROR domain.

It’s not really reasonable to attempt to handle failures of this call. The only reasonable thing to do when an error is detected is to call `g_error()`.

Parameters

builder	a GtkBuilder
resource_path	the path of the resource file to parse
error	return location for an error, or `NULL`. [allow-none]

Returns

A positive value on success, 0 if an error occurred

Since: 3.4

---

gtk_builder_add_from_string ()

```
guint
gtk_builder_add_from_string (GtkBuilder *builder,
                             const gchar *buffer,
                             gsize length,
                             GError **error);
```

Parses a string containing a GtkBuilder UI definition and merges it with the current contents of `builder` .

Most users will probably want to use `gtk_builder_new_from_string()`.

Upon errors 0 will be returned and `error` will be assigned a GError from the GTK_BUILDER_ERROR, G_MARKUP_ERROR or G_VARIANT_PARSE_ERROR domain.

It’s not really reasonable to attempt to handle failures of this call. The only reasonable thing to do when an error is detected is to call `g_error()`.

Parameters

builder	a GtkBuilder
buffer	the string to parse
length	the length of `buffer` (may be -1 if `buffer` is nul-terminated)
error	return location for an error, or `NULL`. [allow-none]

Returns

A positive value on success, 0 if an error occurred

Since: 2.12

gtk_builder_add_objects_from_file ()

```
guint
gtk_builder_add_objects_from_file (GtkBuilder *builder,
                                   const gchar *filename,
                                   gchar **object_ids,
                                   GError **error);
```

Parses a file containing a GtkBuilder UI definition building only the requested objects and merges them with the current contents of `builder` .

Upon errors 0 will be returned and `error` will be assigned a GError from the GTK_BUILDER_ERROR, G_MARKUP_ERROR or G_FILE_ERROR domain.

If you are adding an object that depends on an object that is not its child (for instance a GtkTreeView that depends on its GtkTreeModel), you have to explicitly list all of them in `object_ids` .

Parameters

builder	a GtkBuilder	
filename	the name of the file to parse	
object_ids	nul-terminated array of objects to build.	[array zero-terminated=1][element-type utf8]
error	return location for an error, or `NULL`.	[allow-none]

Returns

A positive value on success, 0 if an error occurred

Since: 2.14

gtk_builder_add_objects_from_string ()

```
guint
gtk_builder_add_objects_from_string (GtkBuilder *builder,
                                     const gchar *buffer,
                                     gsize length,
                                     gchar **object_ids,
                                     GError **error);
```

Parses a string containing a GtkBuilder UI definition building only the requested objects and merges them with the current contents of `builder` .

Upon errors 0 will be returned and `error` will be assigned a GError from the GTK_BUILDER_ERROR or G_MARKUP_ERROR domain.

If you are adding an object that depends on an object that is not its child (for instance a GtkTreeView that depends on its GtkTreeModel), you have to explicitly list all of them in `object_ids` .

Parameters

builder	a GtkBuilder	
buffer	the string to parse	
length	the length of `buffer` (may be -1 if `buffer` is nul-terminated)	
object_ids	nul-terminated array of objects to build.	[array zero-terminated=1][element-type utf8]
error	return location for an error, or `NULL`.	[allow-none]

Returns

A positive value on success, 0 if an error occurred

Since: 2.14

gtk_builder_add_objects_from_resource ()

```
guint
gtk_builder_add_objects_from_resource (GtkBuilder *builder,
                                       const gchar *resource_path,
                                       gchar **object_ids,
                                       GError **error);
```

Parses a resource file containing a GtkBuilder UI definition building only the requested objects and merges them with the current contents of `builder` .

Upon errors 0 will be returned and `error` will be assigned a GError from the GTK_BUILDER_ERROR, G_MARKUP_ERROR or G_RESOURCE_ERROR domain.

If you are adding an object that depends on an object that is not its child (for instance a GtkTreeView that depends on its GtkTreeModel), you have to explicitly list all of them in `object_ids` .

Parameters

builder	a GtkBuilder	
resource_path	the path of the resource file to parse	
object_ids	nul-terminated array of objects to build.	[array zero-terminated=1][element-type utf8]
error	return location for an error, or `NULL`.	[allow-none]

Returns

A positive value on success, 0 if an error occurred

Since: 3.4

gtk_builder_extend_with_template ()

```
guint
gtk_builder_extend_with_template (GtkBuilder *builder,
                                  GtkWidget *widget,
                                  GType template_type,
                                  const gchar *buffer,
                                  gsize length,
                                  GError **error);
```

Main private entry point for building composite container components from template XML.

This is exported purely to let gtk-builder-tool validate templates, applications have no need to call this function.

Parameters

builder	a GtkBuilder
widget	the widget that is being extended
template_type	the type that the template is for
buffer	the string to parse
length	the length of `buffer` (may be -1 if `buffer` is nul-terminated)
error	return location for an error, or `NULL`. [allow-none]

Returns

A positive value on success, 0 if an error occurred

---

gtk_builder_get_object ()

```
GObject *
gtk_builder_get_object (GtkBuilder *builder,
                        const gchar *name);
```

Gets the object named `name` . Note that this function does not increment the reference count of the returned object.

Parameters

builder	a GtkBuilder
name	name of object to get

Returns

the object named `name` or `NULL` if it could not be found in the object tree.

[nullable][transfer none]

Since: 2.12

---

gtk_builder_get_objects ()

```
GSList *
gtk_builder_get_objects (GtkBuilder *builder);
```

Gets all objects that have been constructed by `builder` . Note that this function does not increment the reference counts of the returned objects.

Parameters

builder	a GtkBuilder

Returns

a newly-allocated GSList containing all the objects constructed by the GtkBuilder instance. It should be freed by `g_slist_free()`.

[element-type GObject][transfer container]

Since: 2.12

---

gtk_builder_expose_object ()

```
void
gtk_builder_expose_object (GtkBuilder *builder,
                           const gchar *name,
                           GObject *object);
```

Add `object` to the `builder` object pool so it can be referenced just like any other object built by builder.

Parameters

builder	a GtkBuilder
name	the name of the object exposed to the builder
object	the object to expose

Since: 3.8

---

gtk_builder_connect_signals ()

```
void
gtk_builder_connect_signals (GtkBuilder *builder,
                             gpointer user_data);
```

This method is a simpler variation of `gtk_builder_connect_signals_full()`. It uses symbols explicitly added to `builder` with prior calls to `gtk_builder_add_callback_symbol()`. In the case that symbols are not explicitly added; it uses GModule's introspective features (by opening the module `NULL`) to look at the application's symbol table. From here it tries to match the signal handler names given in the interface description with symbols in the application and connects the signals. Note that this function can only be called once, subsequent calls will do nothing.

Note that unless `gtk_builder_add_callback_symbol()` is called for all signal callbacks which are referenced by the loaded XML, this function will require that GModule be supported on the platform.

If you rely on GModule support to lookup callbacks in the symbol table, the following details should be noted:

When compiling applications for Windows, you must declare signal callbacks with G_MODULE_EXPORT, or they will not be put in the symbol table. On Linux and Unices, this is not necessary; applications should instead be compiled with the -Wl,--export-dynamic CFLAGS, and linked against gmodule-export-2.0.

Parameters

builder	a GtkBuilder	
user_data	user data to pass back with all signals	

Since: 2.12

---

gtk_builder_connect_signals_full ()

```
void
gtk_builder_connect_signals_full (GtkBuilder *builder,
                                  GtkBuilderConnectFunc func,
                                  gpointer user_data);
```

This function can be thought of the interpreted language binding version of `gtk_builder_connect_signals()`, except that it does not require GModule to function correctly.

Parameters

builder	a GtkBuilder	
func	the function used to connect the signals.	[scope call]
user_data	arbitrary data that will be passed to the connection function	

Since: 2.12

---

gtk_builder_set_translation_domain ()

```
void
gtk_builder_set_translation_domain (GtkBuilder *builder,
                                    const gchar *domain);
```

Sets the translation domain of `builder` . See “translation-domain”.

Parameters

builder	a GtkBuilder	
domain	the translation domain or `NULL`.	[allow-none]

Since: 2.12

---

gtk_builder_get_translation_domain ()

```
const gchar *
gtk_builder_get_translation_domain (GtkBuilder *builder);
```

Gets the translation domain of `builder` .

Parameters

builder	a GtkBuilder

Returns

the translation domain. This string is owned by the builder object and must not be modified or freed.

Since: 2.12

---

gtk_builder_set_application ()

```
void
gtk_builder_set_application (GtkBuilder *builder,
                             GtkApplication *application);
```

Sets the application associated with `builder` .

You only need this function if there is more than one GApplication in your process. `application` cannot be `NULL`.

Parameters

builder	a GtkBuilder
application	a GtkApplication

Since: 3.10

---

gtk_builder_get_application ()

```
GtkApplication *
gtk_builder_get_application (GtkBuilder *builder);
```

Gets the GtkApplication associated with the builder.

The GtkApplication is used for creating action proxies as requested from XML that the builder is loading.

By default, the builder uses the default application: the one from `g_application_get_default()`. If you want to use another application for constructing proxies, use `gtk_builder_set_application()`.

Parameters

builder	a GtkBuilder

Returns

the application being used by the builder, or `NULL`.

[nullable][transfer none]

Since: 3.10

---

gtk_builder_get_type_from_name ()

```
GType
gtk_builder_get_type_from_name (GtkBuilder *builder,
                                const char *type_name);
```

Looks up a type by name, using the virtual function that GtkBuilder has for that purpose. This is mainly used when implementing the GtkBuildable interface on a type.

Parameters

builder	a GtkBuilder
type_name	type name to lookup

Returns

the GType found for `type_name` or G_TYPE_INVALID if no type was found

Since: 2.12

---

gtk_builder_value_from_string ()

```
gboolean
gtk_builder_value_from_string (GtkBuilder *builder,
                               GParamSpec *pspec,
                               const gchar *string,
                               GValue *value,
                               GError **error);
```

This function demarshals a value from a string. This function calls `g_value_init()` on the `value` argument, so it need not be initialised beforehand.

This function can handle char, uchar, boolean, int, uint, long, ulong, enum, flags, float, double, string, GdkColor, GdkRGBA and GtkAdjustment type values. Support for GtkWidget type values is still to come.

Upon errors `FALSE` will be returned and `error` will be assigned a GError from the GTK_BUILDER_ERROR domain.

Parameters

builder	a GtkBuilder	
pspec	the GParamSpec for the property	
string	the string representation of the value	
value	the GValue to store the result in.	[out]
error	return location for an error, or `NULL`.	[allow-none]

Returns

`TRUE` on success

Since: 2.12

gtk_builder_value_from_string_type ()

```
gboolean
gtk_builder_value_from_string_type (GtkBuilder *builder,
                                    GType type,
                                    const gchar *string,
                                    GValue *value,
                                    GError **error);
```

Like `gtk_builder_value_from_string()`, this function demarshals a value from a string, but takes a GType instead of GParamSpec. This function calls `g_value_init()` on the `value` argument, so it need not be initialised beforehand.

Upon errors `FALSE` will be returned and `error` will be assigned a GError from the GTK_BUILDER_ERROR domain.

Parameters

builder	a GtkBuilder	
type	the GType of the value	
string	the string representation of the value	
value	the GValue to store the result in.	[out]
error	return location for an error, or `NULL`.	[allow-none]

Returns

`TRUE` on success

Since: 2.12

GTK_BUILDER_WARN_INVALID_CHILD_TYPE()

```
#define             GTK_BUILDER_WARN_INVALID_CHILD_TYPE(object, type)
```

This macro should be used to emit a warning about and unexpected `type` value in a GtkBuildable add_child implementation.

Parameters

object	the GtkBuildable on which the warning ocurred
type	the unexpected type value

Types and Values

GtkBuilder

```
typedef struct _GtkBuilder GtkBuilder;
```

---

enum GtkBuilderError

Error codes that identify various errors that can occur while using GtkBuilder.

Members

GTK_BUILDER_ERROR_INVALID_TYPE_FUNCTION	A type-func attribute didn’t name a function that returns a GType.
GTK_BUILDER_ERROR_UNHANDLED_TAG	The input contained a tag that GtkBuilder can’t handle.
GTK_BUILDER_ERROR_MISSING_ATTRIBUTE	An attribute that is required by GtkBuilder was missing.
GTK_BUILDER_ERROR_INVALID_ATTRIBUTE	GtkBuilder found an attribute that it doesn’t understand.
GTK_BUILDER_ERROR_INVALID_TAG	GtkBuilder found a tag that it doesn’t understand.
GTK_BUILDER_ERROR_MISSING_PROPERTY_VALUE	A required property value was missing.
GTK_BUILDER_ERROR_INVALID_VALUE	GtkBuilder couldn’t parse some attribute value.
GTK_BUILDER_ERROR_VERSION_MISMATCH	The input file requires a newer version of GTK+.
GTK_BUILDER_ERROR_DUPLICATE_ID	An object id occurred twice.
GTK_BUILDER_ERROR_OBJECT_TYPE_REFUSED	A specified object type is of the same type or derived from the type of the composite class being extended with builder XML.
GTK_BUILDER_ERROR_TEMPLATE_MISMATCH	The wrong type was specified in a composite class’s template XML
GTK_BUILDER_ERROR_INVALID_PROPERTY	The specified property is unknown for the object class.
GTK_BUILDER_ERROR_INVALID_SIGNAL	The specified signal is unknown for the object class.
GTK_BUILDER_ERROR_INVALID_ID	An object id is unknown

GTK_BUILDER_ERROR

```
#define GTK_BUILDER_ERROR               (gtk_builder_error_quark ())
```

Property Details

The “translation-domain” property

```
  “translation-domain”       gchar *
```

The translation domain used when translating property values that have been marked as translatable in interface descriptions. If the translation domain is `NULL`, GtkBuilder uses `gettext()`, otherwise `g_dgettext()`.

Flags: Read / Write

Default value: NULL

Since: 2.12

GtkBuildable

GtkBuildable — Interface for objects that can be built by GtkBuilder

Functions

void	gtk_buildable_set_name ()
const gchar *	gtk_buildable_get_name ()
void	gtk_buildable_add_child ()
void	gtk_buildable_set_buildable_property ()
GObject *	gtk_buildable_construct_child ()
gboolean	gtk_buildable_custom_tag_start ()
void	gtk_buildable_custom_tag_end ()
void	gtk_buildable_custom_finished ()
void	gtk_buildable_parser_finished ()
GObject *	gtk_buildable_get_internal_child ()

Types and Values

	GtkBuildable
struct	GtkBuildableIface

Object Hierarchy

```
GInterface
└── GtkBuildable
```

Prerequisites

GtkBuildable requires GObject.

Known Implementations

GtkBuildable is implemented by GtkAboutDialog, GtkAccelLabel, GtkAction, GtkActionBar, GtkActionGroup, GtkAlignment, GtkAppChooserButton, GtkAppChooserDialog, GtkAppChooserWidget, GtkApplicationWindow, GtkArrow, GtkAspectFrame, GtkAssistant, GtkBin, GtkBox, GtkButton, GtkButtonBox, GtkCalendar, GtkCellArea, GtkCellAreaBox, GtkCellView, GtkCheckButton, GtkCheckMenuItem, GtkColorButton, GtkColorChooserDialog, GtkColorChooserWidget, GtkColorSelection, GtkColorSelectionDialog, GtkComboBox, GtkComboBoxText, GtkContainer, GtkDialog, GtkDrawingArea, GtkEntry, GtkEntryCompletion, GtkEventBox, GtkExpander, GtkFileChooserButton, GtkFileChooserDialog, GtkFileChooserWidget, GtkFileFilter, GtkFixed, GtkFlowBox, GtkFlowBoxChild, GtkFontButton, GtkFontChooserDialog, GtkFontChooserWidget, GtkFontSelection, GtkFontSelectionDialog, GtkFrame, GtkGLArea, GtkGrid, GtkHBox, GtkHButtonBox, GtkHPaned, GtkHSV, GtkHScale, GtkHScrollbar, GtkHSeparator, GtkHandleBox, GtkHeaderBar, GtkIconFactory, GtkIconView, GtkImage, GtkImageMenuItem, GtkInfoBar, GtkInvisible, GtkLabel, GtkLayout, GtkLevelBar, GtkLinkButton, GtkListBox, GtkListBoxRow, GtkListStore, GtkLockButton, GtkMenu, GtkMenuBar, GtkMenuButton, GtkMenuItem, GtkMenuShell, GtkMenuToolButton, GtkMessageDialog, GtkMisc, GtkModelButton, GtkNotebook, GtkOffscreenWindow, GtkOverlay, GtkPageSetupUnixDialog, GtkPaned, GtkPlacesSidebar, GtkPlug, GtkPopover, GtkPopoverMenu, GtkPrintUnixDialog, GtkProgressBar, GtkRadioAction, GtkRadioButton, GtkRadioMenuItem, GtkRadioToolButton, GtkRange, GtkRecentAction, GtkRecentChooserDialog, GtkRecentChooserMenu, GtkRecentChooserWidget, GtkRecentFilter, GtkRevealer, GtkScale, GtkScaleButton, GtkScrollbar, GtkScrolledWindow, GtkSearchBar, GtkSearchEntry, GtkSeparator, GtkSeparatorMenuItem, GtkSeparatorToolItem, GtkShortcutsGroup, GtkShortcutsSection, GtkShortcutsShortcut, GtkShortcutsWindow, GtkSizeGroup, GtkSocket, GtkSpinButton, GtkSpinner, GtkStack, GtkStackSidebar, GtkStackSwitcher, GtkStatusbar, GtkSwitch, GtkTable, GtkTearoffMenuItem, GtkTextTagTable, GtkTextView, GtkToggleAction, GtkToggleButton, GtkToggleToolButton, GtkToolButton, GtkToolItem, GtkToolItemGroup, GtkToolPalette, GtkToolbar, GtkTreeStore, GtkTreeView, GtkTreeViewColumn, GtkUIManager, GtkVBox, GtkVButtonBox, GtkVPaned, GtkVScale, GtkVScrollbar, GtkVSeparator, GtkViewport, GtkVolumeButton, GtkWidget and GtkWindow.

Includes

```
#include <gtk/gtk.h>
```

Description

GtkBuildable allows objects to extend and customize their deserialization from GtkBuilder UI descriptions. The interface includes methods for setting names and properties of objects, parsing custom tags and constructing child objects.

The GtkBuildable interface is implemented by all widgets and many of the non-widget objects that are provided by GTK+. The main user of this interface is GtkBuilder. There should be very little need for applications to call any of these functions directly.

An object only needs to implement this interface if it needs to extend the GtkBuilder format or run any extra routines at deserialization time.

Functions

gtk_buildable_set_name ()

```
void
gtk_buildable_set_name (GtkBuildable *buildable,
                        const gchar *name);
```

Sets the name of the `buildable` object.

Parameters

buildable	a GtkBuildable
name	name to set

Since: 2.12

---

gtk_buildable_get_name ()

```
const gchar *
gtk_buildable_get_name (GtkBuildable *buildable);
```

Gets the name of the `buildable` object.

GtkBuilder sets the name based on the GtkBuilder UI definition used to construct the `buildable` .

Parameters

buildable	a GtkBuildable

Returns

the name set with `gtk_buildable_set_name()`

Since: 2.12

---

gtk_buildable_add_child ()

```
void
gtk_buildable_add_child (GtkBuildable *buildable,
                         GtkBuilder *builder,
                         GObject *child,
                         const gchar *type);
```

Adds a child to `buildable` . `type` is an optional string describing how the child should be added.

Parameters

buildable	a GtkBuildable	
builder	a GtkBuilder	
child	child to add	
type	kind of child or `NULL`.	[allow-none]

Since: 2.12

---

gtk_buildable_set_buildable_property ()

```
void
gtk_buildable_set_buildable_property (GtkBuildable *buildable,
                                      GtkBuilder *builder,
                                      const gchar *name,
                                      const GValue *value);
```

Sets the property name `name` to `value` on the `buildable` object.

Parameters

buildable	a GtkBuildable
builder	a GtkBuilder
name	name of property
value	value of property

Since: 2.12

---

gtk_buildable_construct_child ()

```
GObject *
gtk_buildable_construct_child (GtkBuildable *buildable,
                               GtkBuilder *builder,
                               const gchar *name);
```

Constructs a child of `buildable` with the name `name` .

GtkBuilder calls this function if a “constructor” has been specified in the UI definition.

Parameters

buildable	A GtkBuildable	
builder	GtkBuilder used to construct this object	
name	name of child to construct	

Returns

the constructed child.

[transfer full]

Since: 2.12

---

gtk_buildable_custom_tag_start ()

```
gboolean
gtk_buildable_custom_tag_start (GtkBuildable *buildable,
                                GtkBuilder *builder,
                                GObject *child,
                                const gchar *tagname,
                                GMarkupParser *parser,
                                gpointer *data);
```

This is called for each unknown element under <child>.

Parameters

buildable	a GtkBuildable	
builder	a GtkBuilder used to construct this object	
child	child object or `NULL` for non-child tags.	[allow-none]
tagname	name of tag	
parser	a GMarkupParser to fill in.	[out]
data	return location for user data that will be passed in to parser functions.	[out]

Returns

`TRUE` if a object has a custom implementation, `FALSE` if it doesn't.

Since: 2.12

---

gtk_buildable_custom_tag_end ()

```
void
gtk_buildable_custom_tag_end (GtkBuildable *buildable,
                              GtkBuilder *builder,
                              GObject *child,
                              const gchar *tagname,
                              gpointer *data);
```

This is called at the end of each custom element handled by the buildable.

Parameters

buildable	A GtkBuildable	
builder	GtkBuilder used to construct this object	
child	child object or `NULL` for non-child tags.	[allow-none]
tagname	name of tag	
data	user data that will be passed in to parser functions.	[type gpointer]

Since: 2.12

---

gtk_buildable_custom_finished ()

```
void
gtk_buildable_custom_finished (GtkBuildable *buildable,
                               GtkBuilder *builder,
                               GObject *child,
                               const gchar *tagname,
                               gpointer data);
```

This is similar to `gtk_buildable_parser_finished()` but is called once for each custom tag handled by the `buildable`.

Parameters

buildable	a GtkBuildable	
builder	a GtkBuilder	
child	child object or `NULL` for non-child tags.	[allow-none]
tagname	the name of the tag	
data	user data created in custom_tag_start	

Since: 2.12

---

gtk_buildable_parser_finished ()

```
void
gtk_buildable_parser_finished (GtkBuildable *buildable,
                               GtkBuilder *builder);
```

Called when the builder finishes the parsing of a GtkBuilder UI definition. Note that this will be called once for each time `gtk_builder_add_from_file()` or `gtk_builder_add_from_string()` is called on a builder.

Parameters

buildable	a GtkBuildable
builder	a GtkBuilder

Since: 2.12

---

gtk_buildable_get_internal_child ()

```
GObject *
gtk_buildable_get_internal_child (GtkBuildable *buildable,
                                  GtkBuilder *builder,
                                  const gchar *childname);
```

Get the internal child called `childname` of the `buildable` object.

Parameters

buildable	a GtkBuildable
builder	a GtkBuilder
childname	name of child

Returns

the internal child of the buildable object.

[transfer none]

Since: 2.12

Types and Values

GtkBuildable

```
typedef struct _GtkBuildable GtkBuildable;
```

---

struct GtkBuildableIface

```
struct GtkBuildableIface {
  GTypeInterface g_iface;

  /* virtual table */
  void          (* set_name)              (GtkBuildable  *buildable,
                                           const gchar   *name);
  const gchar * (* get_name)              (GtkBuildable  *buildable);
  void          (* add_child)             (GtkBuildable  *buildable,
                                           GtkBuilder    *builder,
                                           GObject       *child,
                                           const gchar   *type);
  void          (* set_buildable_property) (GtkBuildable  *buildable,
                                           GtkBuilder    *builder,
                                           const gchar   *name,
                                           const GValue  *value);
  GObject *     (* construct_child)        (GtkBuildable  *buildable,
                                           GtkBuilder    *builder,
                                           const gchar   *name);
  gboolean      (* custom_tag_start)       (GtkBuildable  *buildable,
                                           GtkBuilder    *builder,
                                           GObject       *child,
                                           const gchar   *tagname,
                                           GMarkupParser *parser,
                                           gpointer      *data);
  void          (* custom_tag_end)         (GtkBuildable  *buildable,
                                           GtkBuilder    *builder,
                                           GObject       *child,
                                           const gchar   *tagname,
                                           gpointer      *data);
  void          (* custom_finished)        (GtkBuildable  *buildable,
                                           GtkBuilder    *builder,
                                           GObject       *child,
                                           const gchar   *tagname,
                                           gpointer       data);
  void          (* parser_finished)        (GtkBuildable  *buildable,
                                           GtkBuilder    *builder);

  GObject *     (* get_internal_child)     (GtkBuildable  *buildable,
                                           GtkBuilder    *builder,
                                           const gchar   *childname);
};
```

The GtkBuildableIface interface contains method that are necessary to allow GtkBuilder to construct an object from a GtkBuilder UI definition.

Members

`set_name` ()	Stores the name attribute given in the GtkBuilder UI definition. GtkWidget stores the name as object data. Implement this method if your object has some notion of "name" and it makes sense to map the XML name attribute to it.
`get_name` ()	The getter corresponding to `set_name` . Implement this if you implement `set_name` .
`add_child` ()	Adds a child. The `type` parameter can be used to differentiate the kind of child. GtkContainer implements this to add add a child widget to the container, GtkNotebook uses the `type` to distinguish between page labels (of type "page-label") and normal children.
`set_buildable_property` ()	Sets a property of a buildable object. It is normally not necessary to implement this, `g_object_set_property()` is used by default. GtkWindow implements this to delay showing itself (i.e. setting the "visible" property) until the whole interface is created.
`construct_child` ()	Constructs a child of a buildable that has been specified as "constructor" in the UI definition. GtkUIManager implements this to reference to a widget created in a <ui> tag which is outside of the normal GtkBuilder UI definition hierarchy. A reference to the constructed object is returned and becomes owned by the caller.
`custom_tag_start` ()	Implement this if the buildable needs to parse content below <child>. To handle an element, the implementation must fill in the `parser` and `user_data` and return `TRUE`. GtkWidget implements this to parse keyboard accelerators specified in <accelerator> elements. GtkContainer implements it to map properties defined via <packing> elements to child properties. Note that `user_data` must be freed in `custom_tag_end` or `custom_finished` .

`custom_tag_end` ()	Called for the end tag of each custom element that is handled by the buildable (see `custom_tag_start`).
`custom_finished` ()	Called for each custom tag handled by the buildable when the builder finishes parsing (see `custom_tag_start`)
`parser_finished` ()	Called when a builder finishes the parsing of a UI definition. It is normally not necessary to implement this, unless you need to perform special cleanup actions. GtkWindow sets the "visible" property here.
`get_internal_child` ()	Returns an internal child of a buildable. GtkDialog implements this to give access to its `vbox` , making it possible to add children to the vbox in a UI definition. Implement this if the buildable has internal children that may need to be accessed from a UI definition.

Windows

GtkWindow — Toplevel which can contain other widgets

GtkDialog — Create popup windows

GtkMessageDialog — A convenient message window

GtkAboutDialog — Display information about an application

GtkAssistant — A widget used to guide users through multi-step operations

GtkInvisible — A widget which is not displayed

GtkOffscreenWindow — A toplevel to manage offscreen rendering of child widgets

GtkWindowGroup — Limit the effect of grabs

GtkWindow

GtkWindow — Toplevel which can contain other widgets

Functions

GtkWidget *	gtk_window_new ()
void	gtk_window_set_title ()
void	gtk_window_set_wmclass ()
void	gtk_window_set_resizable ()
gboolean	gtk_window_get_resizable ()
void	gtk_window_add_accel_group ()
void	gtk_window_remove_accel_group ()
gboolean	gtk_window_activate_focus ()
gboolean	gtk_window_activate_default ()
void	gtk_window_set_modal ()
void	gtk_window_set_default_size ()
void	gtk_window_set_default_geometry ()
void	gtk_window_set_geometry_hints ()
void	gtk_window_set_gravity ()
GdkGravity	gtk_window_get_gravity ()
void	gtk_window_set_position ()
void	gtk_window_set_transient_for ()
void	gtk_window_set_attached_to ()
void	gtk_window_set_destroy_with_parent ()
void	gtk_window_set_hide_titlebar_when_maximized ()
void	gtk_window_set_screen ()
GdkScreen *	gtk_window_get_screen ()
gboolean	gtk_window_is_active ()
gboolean	gtk_window_is_maximized ()
gboolean	gtk_window_has_toplevel_focus ()
GList *	gtk_window_list_toplevels ()
void	gtk_window_add_mnemonic ()
void	gtk_window_remove_mnemonic ()
gboolean	gtk_window_mnemonic_activate ()
gboolean	gtk_window_activate_key ()
gboolean	gtk_window_propagate_key_event ()
GtkWidget *	gtk_window_get_focus ()

void	gtk_window_set_focus ()
GtkWidget *	gtk_window_get_default_widget ()
void	gtk_window_set_default ()
void	gtk_window_present ()
void	gtk_window_present_with_time ()
void	gtk_window_close ()
void	gtk_window_iconify ()
void	gtk_window_deiconify ()
void	gtk_window_stick ()
void	gtk_window_unstick ()
void	gtk_window_maximize ()
void	gtk_window_unmaximize ()
void	gtk_window_fullscreen ()
void	gtk_window_fullscreen_on_monitor ()
void	gtk_window_unfullscreen ()
void	gtk_window_set_keep_above ()
void	gtk_window_set_keep_below ()
void	gtk_window_begin_resize_drag ()
void	gtk_window_begin_move_drag ()
void	gtk_window_set_decorated ()
void	gtk_window_set_deletable ()
void	gtk_window_set_mnemonic_modifier ()
void	gtk_window_set_type_hint ()
void	gtk_window_set_skip_taskbar_hint ()
void	gtk_window_set_skip_pager_hint ()
void	gtk_window_set_urgency_hint ()
void	gtk_window_set_accept_focus ()
void	gtk_window_set_focus_on_map ()
void	gtk_window_set_startup_id ()
void	gtk_window_set_role ()
gboolean	gtk_window_get_decorated ()
gboolean	gtk_window_get_deletable ()
GList *	gtk_window_get_default_icon_list ()
const gchar *	gtk_window_get_default_icon_name ()
void	gtk_window_get_default_size ()
gboolean	gtk_window_get_destroy_with_parent ()
gboolean	gtk_window_get_hide_titlebar_when_maximized ()
GdkPixbuf *	gtk_window_get_icon ()
GList *	gtk_window_get_icon_list ()
const gchar *	gtk_window_get_icon_name ()
GdkModifierType	gtk_window_get_mnemonic_modifier ()
gboolean	gtk_window_get_modal ()
void	gtk_window_get_position ()
const gchar *	gtk_window_get_role ()
void	gtk_window_get_size ()
const gchar *	gtk_window_get_title ()
GtkWindow *	gtk_window_get_transient_for ()
GtkWidget *	gtk_window_get_attached_to ()
GdkWindowTypeHint	gtk_window_get_type_hint ()
gboolean	gtk_window_get_skip_taskbar_hint ()
gboolean	gtk_window_get_skip_pager_hint ()
gboolean	gtk_window_get_urgency_hint ()

gboolean	gtk_window_get_accept_focus ()
gboolean	gtk_window_get_focus_on_map ()
GtkWindowGroup *	gtk_window_get_group ()
gboolean	gtk_window_has_group ()
GtkWindowType	gtk_window_get_window_type ()
void	gtk_window_move ()
gboolean	gtk_window_parse_geometry ()
void	gtk_window_reshow_with_initial_size ()
void	gtk_window_resize ()
void	gtk_window_resize_to_geometry ()
void	gtk_window_set_default_icon_list ()
void	gtk_window_set_default_icon ()
gboolean	gtk_window_set_default_icon_from_file ()
void	gtk_window_set_default_icon_name ()
void	gtk_window_set_icon ()
void	gtk_window_set_icon_list ()
gboolean	gtk_window_set_icon_from_file ()
void	gtk_window_set_icon_name ()
void	gtk_window_set_auto_startup_notification ()
gdouble	gtk_window_get_opacity ()
void	gtk_window_set_opacity ()
gboolean	gtk_window_get_mnemonics_visible ()
void	gtk_window_set_mnemonics_visible ()
gboolean	gtk_window_get_focus_visible ()
void	gtk_window_set_focus_visible ()
void	gtk_window_set_has_resize_grip ()
gboolean	gtk_window_get_has_resize_grip ()
gboolean	gtk_window_resize_grip_is_visible ()
gboolean	gtk_window_get_resize_grip_area ()
GtkApplication *	gtk_window_get_application ()
void	gtk_window_set_application ()
void	gtk_window_set_has_user_ref_count ()
void	gtk_window_set_titlebar ()
GtkWidget *	gtk_window_get_titlebar ()
void	gtk_window_set_interactive_debugging ()

Properties

gboolean	accept-focus	Read / Write
GtkApplication *	application	Read / Write
GtkWidget *	attached-to	Read / Write / Construct
gboolean	decorated	Read / Write
gint	default-height	Read / Write
gint	default-width	Read / Write
gboolean	deletable	Read / Write
gboolean	destroy-with-parent	Read / Write
gboolean	focus-on-map	Read / Write
gboolean	focus-visible	Read / Write
GdkGravity	gravity	Read / Write
gboolean	has-resize-grip	Read / Write
gboolean	has-toplevel-focus	Read

gboolean	hide-titlebar-when-maximized	Read / Write
GdkPixbuf *	icon	Read / Write
gchar *	icon-name	Read / Write
gboolean	is-active	Read
gboolean	is-maximized	Read
gboolean	mnemonics-visible	Read / Write
gboolean	modal	Read / Write
gboolean	resizable	Read / Write
gboolean	resize-grip-visible	Read
gchar *	role	Read / Write
GdkScreen *	screen	Read / Write
gboolean	skip-pager-hint	Read / Write
gboolean	skip-taskbar-hint	Read / Write
gchar *	startup-id	Write
gchar *	title	Read / Write
GtkWindow *	transient-for	Read / Write / Construct
GtkWindowType	type	Read / Write / Construct Only
GdkWindowTypeHint	type-hint	Read / Write
gboolean	urgency-hint	Read / Write
GtkWindowPosition	window-position	Read / Write

Style Properties

gchar *	decoration-button-layout	Read
gint	decoration-resize-handle	Read / Write

Signals

void	activate-default	Action
void	activate-focus	Action
gboolean	enable-debugging	Action
void	keys-changed	Run First
void	set-focus	Run Last

Types and Values

	GtkWindow
struct	GtkWindowClass
enum	GtkWindowType
enum	GtkWindowPosition

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkContainer
            └── GtkBin
                └── GtkWindow
                    ├── GtkDialog
                    ├── GtkApplicationWindow
                    ├── GtkAssistant
                    ├── GtkOffscreenWindow
                    ├── GtkPlug
                    └── GtkShortcutsWindow
```

Implemented Interfaces

GtkWindow implements AtkImplementorIface and GtkBuildable.

Includes

```
#include <gtk/gtk.h>
```

Description

A GtkWindow is a toplevel window which can contain other widgets. Windows normally have decorations that are under the control of the windowing system and allow the user to manipulate the window (resize it, move it, close it,...).

GtkWindow as GtkBuildable

The GtkWindow implementation of the GtkBuildable interface supports a custom <accel-groups> element, which supports any number of <group> elements representing the GtkAccelGroup objects you want to add to your window (synonymous with `gtk_window_add_accel_group()`.

It also supports the <initial-focus> element, whose name property names the widget to receive the focus when the window is mapped.

An example of a UI definition fragment with accel groups:

```
1   <object class="GtkWindow">
2     <accel-groups>
3       <group name="accelgroup1"/>
4     </accel-groups>
5     <initial-focus name="thunderclap"/>
6   </object>
7
8   ...
9
10  <object class="GtkAccelGroup" id="accelgroup1"/>
```

The GtkWindow implementation of the GtkBuildable interface supports setting a child as the titlebar by specifying “titlebar” as the “type” attribute of a <child> element.

CSS nodes

```
1   window.background
2   ├── decoration
3   ├── <titlebar child>.titlebar [.default-decoration]
4   └── <child>
```

GtkWindow has a main CSS node with name window and style class .background, and a subnode with name decoration.

Style classes that are typically used with the main CSS node are .csd (when client-side decorations are in use), .solid-csd (for client-side decorations without invisible borders), .ssd (used by mutter when rendering server-side decorations). GtkWindow also represents window states with the following style classes on the main node: .tiled, .maximized, .fullscreen. Specialized types of window often add their own discriminating style classes, such as .popup or .tooltip.

GtkWindow adds the .titlebar and .default-decoration style classes to the widget that is added as a titlebar child.

Functions

gtk_window_new ()

```
GtkWidget *
gtk_window_new (GtkWindowType type);
```

Creates a new GtkWindow, which is a toplevel window that can contain other widgets. Nearly always, the type of the window should be GTK_WINDOW_TOPLEVEL. If you're implementing something like a popup menu from scratch (which is a bad idea, just use GtkMenu), you might use GTK_WINDOW_POPUP. GTK_WINDOW_POPUP is not for dialogs, though in some other toolkits dialogs are called "popups". In GTK+, GTK_WINDOW_POPUP means a pop-up menu or pop-up tooltip. On X11, popup windows are not controlled by the window manager.

If you simply want an undecorated window (no window borders), use `gtk_window_set_decorated()`, don't use GTK_WINDOW_POPUP.

All top-level windows created by `gtk_window_new()` are stored in an internal top-level window list. This list can be obtained from `gtk_window_list_toplevels()`. Due to Gtk+ keeping a reference to the window internally, `gtk_window_new()` does not return a reference to the caller.

To delete a GtkWindow, call `gtk_widget_destroy()`.

Parameters

type	type of window

Returns

a new GtkWindow.

---

gtk_window_set_title ()

```
void
gtk_window_set_title (GtkWindow *window,
                      const gchar *title);
```

Sets the title of the GtkWindow. The title of a window will be displayed in its title bar; on the X Window System, the title bar is rendered by the window manager, so exactly how the title appears to users may vary according to a user's exact configuration. The title should help a user distinguish this window from other windows they may have open. A good title might include the application name and current document filename, for example.

Parameters

window	a GtkWindow
title	title of the window

---

gtk_window_set_wmclass ()

```
void
gtk_window_set_wmclass (GtkWindow *window,
                        const gchar *wmclass_name,
                        const gchar *wmclass_class);
```

`gtk_window_set_wmclass` has been deprecated since version 3.22 and should not be used in newly-written code.

Don't use this function. It sets the X Window System "class" and "name" hints for a window. According to the ICCCM, you should always set these to the same value for all windows in an application, and GTK+ sets them to that value by default, so calling this function is sort of pointless. However, you may want to call `gtk_window_set_role()` on each window in your application, for the benefit of the session manager. Setting the role allows the window manager to restore window positions when loading a saved session.

Parameters

window	a GtkWindow
wmclass_name	window name hint
wmclass_class	window class hint

---

gtk_window_set_resizable ()

```
void
gtk_window_set_resizable (GtkWindow *window,
                          gboolean resizable);
```

Sets whether the user can resize a window. Windows are user resizable by default.

Parameters

window	a GtkWindow
resizable	`TRUE` if the user can resize this window

---

gtk_window_get_resizable ()

```
gboolean
gtk_window_get_resizable (GtkWindow *window);
```

Gets the value set by `gtk_window_set_resizable()`.

Parameters

window	a GtkWindow

Returns

`TRUE` if the user can resize the window

---

gtk_window_add_accel_group ()

```
void
gtk_window_add_accel_group (GtkWindow *window,
                            GtkAccelGroup *accel_group);
```

Associate `accel_group` with `window` , such that calling `gtk_accel_groups_activate()` on `window` will activate accelerators in `accel_group` .

Parameters

window	window to attach accelerator group to
accel_group	a GtkAccelGroup

---

gtk_window_remove_accel_group ()

```
void
gtk_window_remove_accel_group (GtkWindow *window,
                               GtkAccelGroup *accel_group);
```

Reverses the effects of `gtk_window_add_accel_group()`.

Parameters

window	a GtkWindow
accel_group	a GtkAccelGroup

---

gtk_window_activate_focus ()

```
gboolean
gtk_window_activate_focus (GtkWindow *window);
```

Activates the current focused widget within the window.

Parameters

window	a GtkWindow

Returns

`TRUE` if a widget got activated.

---

gtk_window_activate_default ()

```
gboolean
gtk_window_activate_default (GtkWindow *window);
```

Activates the default widget for the window, unless the current focused widget has been configured to receive the default action (see `gtk_widget_set_receives_default()`), in which case the focused widget is activated.

Parameters

window	a GtkWindow

Returns

`TRUE` if a widget got activated.

---

gtk_window_set_modal ()

```
void
gtk_window_set_modal (GtkWindow *window,
                      gboolean modal);
```

Sets a window modal or non-modal. Modal windows prevent interaction with other windows in the same application. To keep modal dialogs on top of main application windows, use `gtk_window_set_transient_for()` to make the dialog transient for the parent; most window managers will then disallow lowering the dialog below the parent.

Parameters

window	a GtkWindow
modal	whether the window is modal

---

gtk_window_set_default_size ()

```
void
gtk_window_set_default_size (GtkWindow *window,
                             gint width,
                             gint height);
```

Sets the default size of a window. If the window's "natural" size (its size request) is larger than the default, the default will be ignored. More generally, if the default size does not obey the geometry hints for the window (`gtk_window_set_geometry_hints()` can be used to set these explicitly), the default size will be clamped to the nearest permitted size.

Unlike `gtk_widget_set_size_request()`, which sets a size request for a widget and thus would keep users from shrinking the window, this function only sets the initial size, just as if the user had resized the window themselves. Users can still shrink the window again as they normally would. Setting a default size of -1 means to use the "natural" default size (the size request of the window).

For more control over a window's initial size and how resizing works, investigate `gtk_window_set_geometry_hints()`.

For some uses, `gtk_window_resize()` is a more appropriate function. `gtk_window_resize()` changes the current size of the window, rather than the size to be used on initial display. `gtk_window_resize()` always affects the window itself, not the geometry widget.

The default size of a window only affects the first time a window is shown; if a window is hidden and re-shown, it will remember the size it had prior to hiding, rather than using the default size.

Windows can't actually be 0x0 in size, they must be at least 1x1, but passing 0 for `width` and `height` is OK, resulting in a 1x1 default size.

If you use this function to reestablish a previously saved window size, note that the appropriate size to save is the one returned by `gtk_window_get_size()`. Using the window allocation directly will not work in all circumstances and can lead to growing or shrinking windows.

Parameters

window	a GtkWindow
width	width in pixels, or -1 to unset the default width
height	height in pixels, or -1 to unset the default height

gtk_window_set_default_geometry ()

```
void
gtk_window_set_default_geometry (GtkWindow *window,
                                 gint width,
                                 gint height);
```

`gtk_window_set_default_geometry` has been deprecated since version 3.20 and should not be used in newly-written code.

This function does nothing. If you want to set a default size, use `gtk_window_set_default_size()` instead.

Like `gtk_window_set_default_size()`, but `width` and `height` are interpreted in terms of the base size and increment set with gtk_window_set_geometry_hints.

Parameters

window	a GtkWindow	
width	width in resize increments, or -1 to unset the default width	
height	height in resize increments, or -1 to unset the default height	

Since: 3.0

---

gtk_window_set_geometry_hints ()

```
void
gtk_window_set_geometry_hints (GtkWindow *window,
                               GtkWidget *geometry_widget,
                               GdkGeometry *geometry,
                               GdkWindowHints geom_mask);
```

This function sets up hints about how a window can be resized by the user. You can set a minimum and maximum size; allowed resize increments (e.g. for xterm, you can only resize by the size of a character); aspect ratios; and more. See the GdkGeometry struct.

Parameters

window	a GtkWindow	
geometry_widget	widget the geometry hints used to be applied to or `NULL`. Since 3.20 this argument is ignored and GTK behaves as if `NULL` was set.	[allow-none]
geometry	struct containing geometry information or `NULL`.	[allow-none]
geom_mask	mask indicating which struct fields should be paid attention to	

---

gtk_window_set_gravity ()

```
void
gtk_window_set_gravity (GtkWindow *window,
                        GdkGravity gravity);
```

Window gravity defines the meaning of coordinates passed to `gtk_window_move()`. See `gtk_window_move()` and GdkGravity for more details.

The default window gravity is GDK_GRAVITY_NORTH_WEST which will typically “do what you mean.”

Parameters

window	a GtkWindow
gravity	window gravity

---

gtk_window_get_gravity ()

```
GdkGravity
gtk_window_get_gravity (GtkWindow *window);
```

Gets the value set by `gtk_window_set_gravity()`.

Parameters

window	a GtkWindow

Returns

window gravity.

[transfer none]

---

gtk_window_set_position ()

```
void
gtk_window_set_position (GtkWindow *window,
                         GtkWindowPosition position);
```

Sets a position constraint for this window. If the old or new constraint is `GTK_WIN_POS_CENTER_ALWAYS`, this will also cause the window to be repositioned to satisfy the new constraint.

Parameters

window	a GtkWindow.
position	a position constraint.

---

gtk_window_set_transient_for ()

```
void
gtk_window_set_transient_for (GtkWindow *window,
                              GtkWindow *parent);
```

Dialog windows should be set transient for the main application window they were spawned from. This allows window managers to e.g. keep the dialog on top of the main window, or center the dialog over the main window. `gtk_dialog_new_with_buttons()` and other convenience functions in GTK+ will sometimes call `gtk_window_set_transient_for()` on your behalf.

Passing `NULL` for `parent` unsets the current transient window.

On Wayland, this function can also be used to attach a new GTK_WINDOW_POPUP to a GTK_WINDOW_TOPLEVEL parent already mapped on screen so that the GTK_WINDOW_POPUP will be created as a subsurface-based window GDK_WINDOW_SUBSURFACE which can be positioned at will relatively to the GTK_WINDOW_TOPLEVEL surface.

On Windows, this function puts the child window on top of the parent, much as the window manager would have done on X.

Parameters

window	a GtkWindow	
parent	parent window, or `NULL`.	[allow-none]

---

gtk_window_set_attached_to ()

```
void
gtk_window_set_attached_to (GtkWindow *window,
                            GtkWidget *attach_widget);
```

Marks `window` as attached to `attach_widget` . This creates a logical binding between the window and the widget it belongs to, which is used by GTK+ to propagate information such as styling or accessibility to `window` as if it was a children of `attach_widget` .

Examples of places where specifying this relation is useful are for instance a GtkMenu created by a GtkComboBox, a completion popup window created by GtkEntry or a typeahead search entry created by GtkTreeView.

Note that this function should not be confused with `gtk_window_set_transient_for()`, which specifies a window manager relation between two toplevels instead.

Passing `NULL` for `attach_widget` detaches the window.

Parameters

window	a GtkWindow	
attach_widget	a GtkWidget, or `NULL`.	[allow-none]

Since: 3.4

---

gtk_window_set_destroy_with_parent ()

```
void
gtk_window_set_destroy_with_parent (GtkWindow *window,
                                    gboolean setting);
```

If `setting` is `TRUE`, then destroying the transient parent of `window` will also destroy `window` itself. This is useful for dialogs that shouldn’t persist beyond the lifetime of the main window they're associated with, for example.

Parameters

window	a GtkWindow
setting	whether to destroy `window` with its transient parent

---

gtk_window_set_hide_titlebar_when_maximized ()

```
void
gtk_window_set_hide_titlebar_when_maximized
                                (GtkWindow *window,
                                 gboolean setting);
```

If `setting` is `TRUE`, then `window` will request that it’s titlebar should be hidden when maximized. This is useful for windows that don’t convey any information other than the application name in the titlebar, to put the available screen space to better use. If the underlying window system does not support the request, the setting will not have any effect.

Note that custom titlebars set with `gtk_window_set_titlebar()` are not affected by this. The application is in full control of their content and visibility anyway.

Parameters

window	a GtkWindow
setting	whether to hide the titlebar when `window` is maximized

Since: 3.4

---

gtk_window_set_screen ()

```
void
gtk_window_set_screen (GtkWindow *window,
                       GdkScreen *screen);
```

Sets the GdkScreen where the `window` is displayed; if the window is already mapped, it will be unmapped, and then remapped on the new screen.

Parameters

window	a GtkWindow.
screen	a GdkScreen.

Since: 2.2

---

gtk_window_get_screen ()

```
GdkScreen *
gtk_window_get_screen (GtkWindow *window);
```

Returns the GdkScreen associated with `window`.

Parameters

window	a GtkWindow.

Returns

a GdkScreen.

[transfer none]

Since: 2.2

---

gtk_window_is_active ()

```
gboolean
gtk_window_is_active (GtkWindow *window);
```

Returns whether the window is part of the current active toplevel. (That is, the toplevel window receiving keystrokes.) The return value is `TRUE` if the window is active toplevel itself, but also if it is, say, a GtkPlug embedded in the active toplevel. You might use this function if you wanted to draw a widget differently in an active window from a widget in an inactive window. See `gtk_window_has_toplevel_focus()`

Parameters

window	a GtkWindow

Returns

`TRUE` if the window part of the current active window.

Since: 2.4

---

gtk_window_is_maximized ()

```
gboolean
gtk_window_is_maximized (GtkWindow *window);
```

Retrieves the current maximized state of `window`.

Note that since maximization is ultimately handled by the window manager and happens asynchronously to an application request, you shouldn't assume the return value of this function changing immediately (or at all), as an effect of calling `gtk_window_maximize()` or `gtk_window_unmaximize()`.

Parameters

window	a GtkWindow

Returns

whether the window has a maximized state.

Since: 3.12

---

gtk_window_has_toplevel_focus ()

```
gboolean
gtk_window_has_toplevel_focus (GtkWindow *window);
```

Returns whether the input focus is within this GtkWindow. For real toplevel windows, this is identical to `gtk_window_is_active()`, but for embedded windows, like GtkPlug, the results will differ.

Parameters

window	a GtkWindow

Returns

`TRUE` if the input focus is within this GtkWindow

Since: 2.4

---

gtk_window_list_toplevels ()

```
GList *
gtk_window_list_toplevels (void);
```

Returns a list of all existing toplevel windows. The widgets in the list are not individually referenced. If you want to iterate through the list and perform actions involving callbacks that might destroy the widgets, you must call `g_list_foreach (result, (GFunc)g_object_ref, NULL)` first, and then unref all the widgets afterwards.

Returns

list of toplevel widgets.

[element-type GtkWidget][transfer container]

---

gtk_window_add_mnemonic ()

```
void
gtk_window_add_mnemonic (GtkWindow *window,
                         guint keyval,
                         GtkWidget *target);
```

Adds a mnemonic to this window.

Parameters

window	a GtkWindow
keyval	the mnemonic
target	the widget that gets activated by the mnemonic

---

gtk_window_remove_mnemonic ()

```
void
gtk_window_remove_mnemonic (GtkWindow *window,
                            guint keyval,
                            GtkWidget *target);
```

Removes a mnemonic from this window.

Parameters

window	a GtkWindow
keyval	the mnemonic
target	the widget that gets activated by the mnemonic

---

gtk_window_mnemonic_activate ()

```
gboolean
gtk_window_mnemonic_activate (GtkWindow *window,
                              guint keyval,
                              GdkModifierType modifier);
```

Activates the targets associated with the mnemonic.

Parameters

window	a GtkWindow
keyval	the mnemonic
modifier	the modifiers

Returns

TRUE if the activation is done.

---

gtk_window_activate_key ()

```
gboolean
gtk_window_activate_key (GtkWindow *window,
                         GdkEventKey *event);
```

Activates mnemonics and accelerators for this GtkWindow. This is normally called by the default ::key_press_event handler for toplevel windows, however in some cases it may be useful to call this directly when overriding the standard key handling for a toplevel window.

Parameters

window	a GtkWindow
event	a GdkEventKey

Returns

TRUE if a mnemonic or accelerator was found and activated.

Since: 2.4

---

gtk_window_propagate_key_event ()

```
gboolean
gtk_window_propagate_key_event (GtkWindow *window,
                                GdkEventKey *event);
```

Propagate a key press or release event to the focus widget and up the focus container chain until a widget handles `event` . This is normally called by the default ::key_press_event and ::key_release_event handlers for toplevel windows, however in some cases it may be useful to call this directly when overriding the standard key handling for a toplevel window.

Parameters

window	a GtkWindow
event	a GdkEventKey

Returns

`TRUE` if a widget in the focus chain handled the event.

Since: 2.4

---

gtk_window_get_focus ()

```
GtkWidget *
gtk_window_get_focus (GtkWindow *window);
```

Retrieves the current focused widget within the window. Note that this is the widget that would have the focus if the toplevel window focused; if the toplevel window is not focused then `gtk_widget_has_focus (widget)` will not be `TRUE` for the widget.

Parameters

window	a GtkWindow

Returns

the currently focused widget, or `NULL` if there is none.

[nullable][transfer none]

---

gtk_window_set_focus ()

```
void
gtk_window_set_focus (GtkWindow *window,
                      GtkWidget *focus);
```

If `focus` is not the current focus widget, and is focusable, sets it as the focus widget for the window. If `focus` is `NULL`, unsets the focus widget for this window. To set the focus to a particular widget in the toplevel, it is usually more convenient to use `gtk_widget_grab_focus()` instead of this function.

Parameters

window	a GtkWindow	
focus	widget to be the new focus widget, or `NULL` to unset any focus widget for the toplevel window.	[allow-none]

---

gtk_window_get_default_widget ()

```
GtkWidget *
gtk_window_get_default_widget (GtkWindow *window);
```

Returns the default widget for `window` . See `gtk_window_set_default()` for more details.

Parameters

window	a GtkWindow	

Returns

the default widget, or `NULL` if there is none.

[nullable][transfer none]

Since: 2.14

---

gtk_window_set_default ()

```
void
gtk_window_set_default (GtkWindow *window,
                        GtkWidget *default_widget);
```

The default widget is the widget that's activated when the user presses Enter in a dialog (for example). This function sets or unsets the default widget for a GtkWindow. When setting (rather than unsetting) the default widget it's generally easier to call `gtk_widget_grab_default()` on the widget. Before making a widget the default widget, you must call `gtk_widget_set_can_default()` on the widget you'd like to make the default.

Parameters

window	a GtkWindow	
default_widget	widget to be the default, or `NULL` to unset the default widget for the toplevel.	[allow-none]

---

gtk_window_present ()

```
void
gtk_window_present (GtkWindow *window);
```

Presents a window to the user. This function should not be used as when it is called, it is too late to gather a valid timestamp to allow focus stealing prevention to work correctly.

Parameters

window	a GtkWindow	

---

gtk_window_present_with_time ()

```
void
gtk_window_present_with_time (GtkWindow *window,
                              guint32 timestamp);
```

Presents a window to the user. This may mean raising the window in the stacking order, deiconifying it, moving it to the current desktop, and/or giving it the keyboard focus, possibly dependent on the user's platform, window manager, and preferences.

If `window` is hidden, this function calls `gtk_widget_show()` as well.

This function should be used when the user tries to open a window that's already open. Say for example the preferences dialog is currently open, and the user chooses Preferences from the menu a second time; use `gtk_window_present()` to move the already-open dialog where the user can see it.

Presents a window to the user in response to a user interaction. The timestamp should be gathered when the window was requested to be shown (when clicking a link for example), rather than once the window is ready to be shown.

Parameters

window	a GtkWindow
timestamp	the timestamp of the user interaction (typically a button or key press event) which triggered this call

Since: 2.8

gtk_window_close ()

```
void
gtk_window_close (GtkWindow *window);
```

Requests that the window is closed, similar to what happens when a window manager close button is clicked.

This function can be used with close buttons in custom titlebars.

Parameters

window	a GtkWindow

Since: 3.10

---

gtk_window_iconify ()

```
void
gtk_window_iconify (GtkWindow *window);
```

Asks to iconify (i.e. minimize) the specified `window` . Note that you shouldn't assume the window is definitely iconified afterward, because other entities (e.g. the user or window manager) could deiconify it again, or there may not be a window manager in which case iconification isn't possible, etc. But normally the window will end up iconified. Just don't write code that crashes if not.

It's permitted to call this function before showing a window, in which case the window will be iconified before it ever appears onscreen.

You can track iconification via the "window-state-event" signal on GtkWidget.

Parameters

window	a GtkWindow

---

gtk_window_deiconify ()

```
void
gtk_window_deiconify (GtkWindow *window);
```

Asks to deiconify (i.e. unminimize) the specified `window` . Note that you shouldn't assume the window is definitely deiconified afterward, because other entities (e.g. the user or window manager)) could iconify it again before your code which assumes deiconification gets to run.

You can track iconification via the "window-state-event" signal on GtkWidget.

Parameters

window	a GtkWindow

---

gtk_window_stick ()

```
void
gtk_window_stick (GtkWindow *window);
```

Asks to stick `window` , which means that it will appear on all user desktops. Note that you shouldn’t assume the window is definitely stuck afterward, because other entities (e.g. the user or window manager could unstick it again, and some window managers do not support sticking windows. But normally the window will end up stuck. Just don't write code that crashes if not.

It’s permitted to call this function before showing a window.

You can track stickiness via the “window-state-event” signal on GtkWidget.

Parameters

window	a GtkWindow

---

gtk_window_unstick ()

```
void
gtk_window_unstick (GtkWindow *window);
```

Asks to unstick `window` , which means that it will appear on only one of the user’s desktops. Note that you shouldn’t assume the window is definitely unstuck afterward, because other entities (e.g. the user or window manager) could stick it again. But normally the window will end up stuck. Just don’t write code that crashes if not.

You can track stickiness via the “window-state-event” signal on GtkWidget.

Parameters

window	a GtkWindow

---

gtk_window_maximize ()

```
void
gtk_window_maximize (GtkWindow *window);
```

Asks to maximize `window` , so that it becomes full-screen. Note that you shouldn’t assume the window is definitely maximized afterward, because other entities (e.g. the user or window manager) could unmaximize it again, and not all window managers support maximization. But normally the window will end up maximized. Just don’t write code that crashes if not.

It’s permitted to call this function before showing a window, in which case the window will be maximized when it appears onscreen initially.

You can track maximization via the “window-state-event” signal on GtkWidget, or by listening to notifications on the “is-maximized” property.

Parameters

window	a GtkWindow

---

gtk_window_unmaximize ()

```
void
gtk_window_unmaximize (GtkWindow *window);
```

Asks to unmaximize `window` . Note that you shouldn’t assume the window is definitely unmaximized afterward, because other entities (e.g. the user or window manager) could maximize it again, and not all window managers honor requests to unmaximize. But normally the window will end up unmaximized. Just don’t write code that crashes if not.

You can track maximization via the “window-state-event” signal on GtkWidget.

Parameters

window	a GtkWindow

---

gtk_window_fullscreen ()

```
void
gtk_window_fullscreen (GtkWindow *window);
```

Asks to place `window` in the fullscreen state. Note that you shouldn’t assume the window is definitely full screen afterward, because other entities (e.g. the user or window manager) could unfullscreen it again, and not all window managers honor requests to fullscreen windows. But normally the window will end up fullscreen. Just don’t write code that crashes if not.

You can track the fullscreen state via the “window-state-event” signal on GtkWidget.

Parameters

window	a GtkWindow

Since: 2.2

---

gtk_window_fullscreen_on_monitor ()

```
void
gtk_window_fullscreen_on_monitor (GtkWindow *window,
                                  GdkScreen *screen,
                                  gint monitor);
```

Asks to place `window` in the fullscreen state. Note that you shouldn't assume the window is definitely full screen afterward.

You can track the fullscreen state via the "window-state-event" signal on GtkWidget.

Parameters

window	a GtkWindow
screen	a GdkScreen to draw to
monitor	which monitor to go fullscreen on

Since: 3.18

---

gtk_window_unfullscreen ()

```
void
gtk_window_unfullscreen (GtkWindow *window);
```

Asks to toggle off the fullscreen state for `window` . Note that you shouldn’t assume the window is definitely not full screen afterward, because other entities (e.g. the user or window manager) could fullscreen it again, and not all window managers honor requests to unfullscreen windows. But normally the window will end up restored to its normal state. Just don’t write code that crashes if not.

You can track the fullscreen state via the “window-state-event” signal on GtkWidget.

Parameters

window	a GtkWindow

Since: 2.2

gtk_window_set_keep_above ()

```
void
gtk_window_set_keep_above (GtkWindow *window,
                           gboolean setting);
```

Asks to keep `window` above, so that it stays on top. Note that you shouldn't assume the window is definitely above afterward, because other entities (e.g. the user or window manager) could not keep it above, and not all window managers support keeping windows above. But normally the window will end kept above. Just don't write code that crashes if not.

It's permitted to call this function before showing a window, in which case the window will be kept above when it appears onscreen initially.

You can track the above state via the "window-state-event" signal on GtkWidget.

Note that, according to the Extended Window Manager Hints Specification, the above state is mainly meant for user preferences and should not be used by applications e.g. for drawing attention to their dialogs.

Parameters

window	a GtkWindow
setting	whether to keep `window` above other windows

Since: 2.4

gtk_window_set_keep_below ()

```
void
gtk_window_set_keep_below (GtkWindow *window,
                           gboolean setting);
```

Asks to keep `window` below, so that it stays in bottom. Note that you shouldn't assume the window is definitely below afterward, because other entities (e.g. the user or window manager) could not keep it below, and not all window managers support putting windows below. But normally the window will be kept below. Just don't write code that crashes if not.

It's permitted to call this function before showing a window, in which case the window will be kept below when it appears onscreen initially.

You can track the below state via the "window-state-event" signal on GtkWidget.

Note that, according to the Extended Window Manager Hints Specification, the above state is mainly meant for user preferences and should not be used by applications e.g. for drawing attention to their dialogs.

Parameters

window	a GtkWindow
setting	whether to keep `window` below other windows

Since: 2.4

gtk_window_begin_resize_drag ()

```
void
gtk_window_begin_resize_drag (GtkWindow *window,
                              GdkWindowEdge edge,
                              gint button,
                              gint root_x,
                              gint root_y,
                              guint32 timestamp);
```

Starts resizing a window. This function is used if an application has window resizing controls. When GDK can support it, the resize will be done using the standard mechanism for the window manager or windowing system. Otherwise, GDK will try to emulate window resizing, potentially not all that well, depending on the windowing system.

Parameters

window	a GtkWindow
button	mouse button that initiated the drag
edge	position of the resize control
root_x	X position where the user clicked to initiate the drag, in root window coordinates
root_y	Y position where the user clicked to initiate the drag
timestamp	timestamp from the click event that initiated the drag

gtk_window_begin_move_drag ()

```
void
gtk_window_begin_move_drag (GtkWindow *window,
                            gint button,
                            gint root_x,
                            gint root_y,
                            guint32 timestamp);
```

Starts moving a window. This function is used if an application has window movement grips. When GDK can support it, the window movement will be done using the standard mechanism for the window manager or windowing system. Otherwise, GDK will try to emulate window movement, potentially not all that well, depending on the windowing system.

Parameters

window	a GtkWindow
button	mouse button that initiated the drag
root_x	X position where the user clicked to initiate the drag, in root window coordinates
root_y	Y position where the user clicked to initiate the drag
timestamp	timestamp from the click event that initiated the drag

---

gtk_window_set_decorated ()

```
void
gtk_window_set_decorated (GtkWindow *window,
                          gboolean setting);
```

By default, windows are decorated with a title bar, resize controls, etc. Some window managers allow GTK+ to disable these decorations, creating a borderless window. If you set the decorated property to `FALSE` using this function, GTK+ will do its best to convince the window manager not to decorate the window. Depending on the system, this function may not have any effect when called on a window that is already visible, so you should call it before calling `gtk_widget_show()`.

On Windows, this function always works, since there's no window manager policy involved.

Parameters

window	a GtkWindow
setting	`TRUE` to decorate the window

---

gtk_window_set_deletable ()

```
void
gtk_window_set_deletable (GtkWindow *window,
                          gboolean setting);
```

By default, windows have a close button in the window frame. Some window managers allow GTK+ to disable this button. If you set the deletable property to `FALSE` using this function, GTK+ will do its best to convince the window manager not to show a close button. Depending on the system, this function may not have any effect when called on a window that is already visible, so you should call it before calling `gtk_widget_show()`.

On Windows, this function always works, since there's no window manager policy involved.

Parameters

window	a GtkWindow
setting	`TRUE` to decorate the window as deletable

Since: 2.10

---

gtk_window_set_mnemonic_modifier ()

```
void
gtk_window_set_mnemonic_modifier (GtkWindow *window,
                                  GdkModifierType modifier);
```

Sets the mnemonic modifier for this window.

Parameters

window	a GtkWindow
modifier	the modifier mask used to activate mnemonics on this window.

---

gtk_window_set_type_hint ()

```
void
gtk_window_set_type_hint (GtkWindow *window,
                          GdkWindowTypeHint hint);
```

By setting the type hint for the window, you allow the window manager to decorate and handle the window in a way which is suitable to the function of the window in your application.

This function should be called before the window becomes visible.

gtk_dialog_new_with_buttons() and other convenience functions in GTK+ will sometimes call `gtk_window_set_type_hint()` on your behalf.

Parameters

window	a GtkWindow
hint	the window type

---

gtk_window_set_skip_taskbar_hint ()

```
void
gtk_window_set_skip_taskbar_hint (GtkWindow *window,
                                  gboolean setting);
```

Windows may set a hint asking the desktop environment not to display the window in the task bar. This function sets this hint.

Parameters

window	a GtkWindow
setting	`TRUE` to keep this window from appearing in the task bar

Since: 2.2

---

gtk_window_set_skip_pager_hint ()

```
void
gtk_window_set_skip_pager_hint (GtkWindow *window,
                                gboolean setting);
```

Windows may set a hint asking the desktop environment not to display the window in the pager. This function sets this hint. (A "pager" is any desktop navigation tool such as a workspace switcher that displays a thumbnail representation of the windows on the screen.)

Parameters

window	a GtkWindow
setting	TRUE to keep this window from appearing in the pager

Since: 2.2

---

gtk_window_set_urgency_hint ()

```
void
gtk_window_set_urgency_hint (GtkWindow *window,
                             gboolean setting);
```

Windows may set a hint asking the desktop environment to draw the users attention to the window. This function sets this hint.

Parameters

window	a GtkWindow
setting	TRUE to mark this window as urgent

Since: 2.8

---

gtk_window_set_accept_focus ()

```
void
gtk_window_set_accept_focus (GtkWindow *window,
                             gboolean setting);
```

Windows may set a hint asking the desktop environment not to receive the input focus. This function sets this hint.

Parameters

window	a GtkWindow
setting	TRUE to let this window receive input focus

Since: 2.4

---

gtk_window_set_focus_on_map ()

```
void
gtk_window_set_focus_on_map (GtkWindow *window,
                             gboolean setting);
```

Windows may set a hint asking the desktop environment not to receive the input focus when the window is mapped. This function sets this hint.

Parameters

window	a GtkWindow
setting	TRUE to let this window receive input focus on map

Since: 2.6

---

gtk_window_set_startup_id ()

```
void
gtk_window_set_startup_id (GtkWindow *window,
                           const gchar *startup_id);
```

Startup notification identifiers are used by desktop environment to track application startup, to provide user feedback and other features. This function changes the corresponding property on the underlying GdkWindow. Normally, startup identifier is managed automatically and you should only use this function in special cases like transferring focus from other processes. You should use this function before calling `gtk_window_present()` or any equivalent function generating a window map event.

This function is only useful on X11, not with other GTK+ targets.

Parameters

window	a GtkWindow
startup_id	a string with startup-notification identifier

Since: 2.12

---

gtk_window_set_role ()

```
void
gtk_window_set_role (GtkWindow *window,
                     const gchar *role);
```

This function is only useful on X11, not with other GTK+ targets.

In combination with the window title, the window role allows a window manager to identify "the same" window when an application is restarted. So for example you might set the “toolbox” role on your app’s toolbox window, so that when the user restarts their session, the window manager can put the toolbox back in the same place.

If a window already has a unique title, you don’t need to set the role, since the WM can use the title to identify the window when restoring the session.

Parameters

window	a GtkWindow
role	unique identifier for the window to be used when restoring a session

---

gtk_window_get_decorated ()

```
gboolean
gtk_window_get_decorated (GtkWindow *window);
```

Returns whether the window has been set to have decorations such as a title bar via `gtk_window_set_decorated()`.

Parameters

window	a GtkWindow

Returns

`TRUE` if the window has been set to have decorations

---

gtk_window_get_deletable ()

```
gboolean
gtk_window_get_deletable (GtkWindow *window);
```

Returns whether the window has been set to have a close button via `gtk_window_set_deletable()`.

Parameters

window	a GtkWindow

Returns

`TRUE` if the window has been set to have a close button

Since: 2.10

---

gtk_window_get_default_icon_list ()

```
GList *
gtk_window_get_default_icon_list (void);
```

Gets the value set by `gtk_window_set_default_icon_list()`. The list is a copy and should be freed with `g_list_free()`, but the pixbufs in the list have not had their reference count incremented.

Returns

copy of default icon list.

[element-type GdkPixbuf][transfer container]

---

gtk_window_get_default_icon_name ()

```
const gchar *
gtk_window_get_default_icon_name (void);
```

Returns the fallback icon name for windows that has been set with `gtk_window_set_default_icon_name()`. The returned string is owned by GTK+ and should not be modified. It is only valid until the next call to `gtk_window_set_default_icon_name()`.

Returns

the fallback icon name for windows

Since: 2.16

---

gtk_window_get_default_size ()

```
void
gtk_window_get_default_size (GtkWindow *window,
                             gint *width,
                             gint *height);
```

Gets the default size of the window. A value of -1 for the width or height indicates that a default size has not been explicitly set for that dimension, so the “natural” size of the window will be used.

Parameters

window	a GtkWindow	
width	location to store the default width, or `NULL`.	[out][allow-none]
height	location to store the default height, or `NULL`.	[out][allow-none]

---

gtk_window_get_destroy_with_parent ()

```
gboolean
gtk_window_get_destroy_with_parent (GtkWindow *window);
```

Returns whether the window will be destroyed with its transient parent. See `gtk_window_set_destroy_with_parent()`.

Parameters

window	a GtkWindow

Returns

`TRUE` if the window will be destroyed with its transient parent.

---

gtk_window_get_hide_titlebar_when_maximized ()

```
gboolean
gtk_window_get_hide_titlebar_when_maximized
                                (GtkWindow *window);
```

Returns whether the window has requested to have its titlebar hidden when maximized. See `gtk_window_set_hide_titlebar_when_maximized()`.

Parameters

window	a GtkWindow

Returns

`TRUE` if the window has requested to have its titlebar hidden when maximized

Since: 3.4

---

gtk_window_get_icon ()

```
GdkPixbuf *
gtk_window_get_icon (GtkWindow *window);
```

Gets the value set by `gtk_window_set_icon()` (or if you've called `gtk_window_set_icon_list()`, gets the first icon in the icon list).

Parameters

window	a GtkWindow

Returns

icon for window or `NULL` if none.

[transfer none][nullable]

---

gtk_window_get_icon_list ()

```
GList *
gtk_window_get_icon_list (GtkWindow *window);
```

Retrieves the list of icons set by `gtk_window_set_icon_list()`. The list is copied, but the reference count on each member won’t be incremented.

Parameters

window	a GtkWindow

Returns

copy of window’s icon list.

[element-type GdkPixbuf][transfer container]

---

gtk_window_get_icon_name ()

```
const gchar *
gtk_window_get_icon_name (GtkWindow *window);
```

Returns the name of the themed icon for the window, see gtk_window_set_icon_name().

Parameters

window	a GtkWindow

Returns

the icon name or NULL if the window has no themed icon.

[nullable]

Since: 2.6

---

gtk_window_get_mnemonic_modifier ()

```
GdkModifierType
gtk_window_get_mnemonic_modifier (GtkWindow *window);
```

Returns the mnemonic modifier for this window. See gtk_window_set_mnemonic_modifier().

Parameters

window	a GtkWindow

Returns

the modifier mask used to activate mnemonics on this window.

---

gtk_window_get_modal ()

```
gboolean
gtk_window_get_modal (GtkWindow *window);
```

Returns whether the window is modal. See gtk_window_set_modal().

Parameters

window	a GtkWindow

Returns

TRUE if the window is set to be modal and establishes a grab when shown

---

gtk_window_get_position ()

```
void
gtk_window_get_position (GtkWindow *window,
                         gint *root_x,
                         gint *root_y);
```

This function returns the position you need to pass to `gtk_window_move()` to keep `window` in its current position. This means that the meaning of the returned value varies with window gravity. See `gtk_window_move()` for more details.

The reliability of this function depends on the windowing system currently in use. Some windowing systems, such as Wayland, do not support a global coordinate system, and thus the position of the window will always be (0, 0). Others, like X11, do not have a reliable way to obtain the geometry of the decorations of a window if they are provided by the window manager. Additionally, on X11, window manager have been known to mismanage window gravity, which result in windows moving even if you use the coordinates of the current position as returned by this function.

If you haven't changed the window gravity, its gravity will be GDK_GRAVITY_NORTH_WEST. This means that `gtk_window_get_position()` gets the position of the top-left corner of the window manager frame for the window. `gtk_window_move()` sets the position of this same top-left corner.

If a window has gravity GDK_GRAVITY_STATIC the window manager frame is not relevant, and thus `gtk_window_get_position()` will always produce accurate results. However you can't use static gravity to do things like place a window in a corner of the screen, because static gravity ignores the window manager decorations.

Ideally, this function should return appropriate values if the window has client side decorations, assuming that the windowing system supports global coordinates.

In practice, saving the window position should not be left to applications, as they lack enough knowledge of the windowing system and the window manager state to effectively do so. The appropriate way to implement saving the window position is to use a platform-specific protocol, wherever that is available.

Parameters

window	a GtkWindow	
root_x	return location for X coordinate of gravity-determined reference point, or `NULL`.	[out][allow-none]
root_y	return location for Y coordinate of gravity-determined reference point, or `NULL`.	[out][allow-none]

gtk_window_get_role ()

```
const gchar *
gtk_window_get_role (GtkWindow *window);
```

Returns the role of the window. See `gtk_window_set_role()` for further explanation.

Parameters

window	a GtkWindow

Returns

the role of the window if set, or `NULL`. The returned is owned by the widget and must not be modified or freed. [nullable]

---

gtk_window_get_size ()

```
void
gtk_window_get_size (GtkWindow *window,
                     gint *width,
                     gint *height);
```

Obtains the current size of `window` .

If `window` is not visible on screen, this function return the size GTK+ will suggest to the window manager for the initial window size (but this is not reliably the same as the size the window manager will actually select). See: `gtk_window_set_default_size()`.

Depending on the windowing system and the window manager constraints, the size returned by this function may not match the size set using `gtk_window_resize()`; additionally, since `gtk_window_resize()` may be implemented as an asynchronous operation, GTK+ cannot guarantee in any way that this code:

```
1  // width and height are set elsewhere
2  gtk_window_resize (window, width, height);
3
4  int new_width, new_height;
5  gtk_window_get_size (window, &new_width, &new_height);
```

will result in `new_width` and `new_height` matching `width` and `height`, respectively.

This function will return the logical size of the GtkWindow, excluding the widgets used in client side decorations; there is, however, no guarantee that the result will be completely accurate because client side decoration may include widgets that depend on the user preferences and that may not be visibile at the time you call this function.

The dimensions returned by this function are suitable for being stored across sessions; use `gtk_window_set_default_size()` to restore them when before showing the window.

To avoid potential race conditions, you should only call this function in response to a size change notification, for instance inside a handler for the "size-allocate" signal, or inside a handler for the "configure-event" signal:

```
1  static void
2  on_size_allocate (GtkWidget *widget, GtkAllocation *allocation)
3  {
4    int new_width, new_height;
5
6    gtk_window_get_size (GTK_WINDOW (widget), &new_width, &new_height);
7
8    ...
9  }
```

Note that, if you connect to the "size-allocate" signal, you should not use the dimensions of the GtkAllocation passed to the signal handler, as the allocation may contain client side decorations added by GTK+, depending on the windowing system in use.

If you are getting a window size in order to position the window on the screen, you should, instead, simply set the window's semantic type with `gtk_window_set_type_hint()`, which allows the window manager to e.g. center dialogs. Also, if you set the transient parent of dialogs with `gtk_window_set_transient_for()`

window managers will often center the dialog over its parent window. It's much preferred to let the window manager handle these cases rather than doing it yourself, because all apps will behave consistently and according to user or system preferences, if the window manager handles it. Also, the window manager can take into account the size of the window decorations and border that it may add, and of which GTK+ has no knowledge. Additionally, positioning windows in global screen coordinates may not be allowed by the windowing system. For more information, see: `gtk_window_set_position()`.

Parameters

window	a GtkWindow	
width	return location for width, or `NULL`.	[out][nullable]
height	return location for height, or `NULL`.	[out][nullable]

---

gtk_window_get_title ()

```
const gchar *
gtk_window_get_title (GtkWindow *window);
```

Retrieves the title of the window. See `gtk_window_set_title()`.

Parameters

window	a GtkWindow

Returns

the title of the window, or `NULL` if none has been set explicitly. The returned string is owned by the widget and must not be modified or freed.

[nullable]

---

gtk_window_get_transient_for ()

```
GtkWindow *
gtk_window_get_transient_for (GtkWindow *window);
```

Fetches the transient parent for this window. See `gtk_window_set_transient_for()`.

Parameters

window	a GtkWindow

Returns

the transient parent for this window, or `NULL` if no transient parent has been set.

[nullable][transfer none]

---

gtk_window_get_attached_to ()

```
GtkWidget *
gtk_window_get_attached_to (GtkWindow *window);
```

Fetches the attach widget for this window. See `gtk_window_set_attached_to()`.

Parameters

window	a GtkWindow

Returns

the widget where the window is attached, or `NULL` if the window is not attached to any widget.

[nullable][transfer none]

Since: 3.4

---

gtk_window_get_type_hint ()

```
GdkWindowTypeHint
gtk_window_get_type_hint (GtkWindow *window);
```

Gets the type hint for this window. See `gtk_window_set_type_hint()`.

Parameters

window	a GtkWindow

Returns

the type hint for `window` .

---

gtk_window_get_skip_taskbar_hint ()

```
gboolean
gtk_window_get_skip_taskbar_hint (GtkWindow *window);
```

Gets the value set by `gtk_window_set_skip_taskbar_hint()`

Parameters

window	a GtkWindow

Returns

`TRUE` if window shouldn’t be in taskbar

Since: 2.2

---

gtk_window_get_skip_pager_hint ()

```
gboolean
gtk_window_get_skip_pager_hint (GtkWindow *window);
```

Gets the value set by `gtk_window_set_skip_pager_hint()`.

Parameters

window	a GtkWindow

Returns

`TRUE` if window shouldn't be in pager

Since: 2.2

---

gtk_window_get_urgency_hint ()

```
gboolean
gtk_window_get_urgency_hint (GtkWindow *window);
```

Gets the value set by `gtk_window_set_urgency_hint()`

Parameters

window	a GtkWindow

Returns

`TRUE` if window is urgent

Since: 2.8

---

gtk_window_get_accept_focus ()

```
gboolean
gtk_window_get_accept_focus (GtkWindow *window);
```

Gets the value set by `gtk_window_set_accept_focus()`.

Parameters

window	a GtkWindow

Returns

`TRUE` if window should receive the input focus

Since: 2.4

---

gtk_window_get_focus_on_map ()

```
gboolean
gtk_window_get_focus_on_map (GtkWindow *window);
```

Gets the value set by `gtk_window_set_focus_on_map()`.

Parameters

window	a GtkWindow	

Returns

`TRUE` if window should receive the input focus when mapped.

Since: 2.6

---

gtk_window_get_group ()

```
GtkWindowGroup *
gtk_window_get_group (GtkWindow *window);
```

Returns the group for `window` or the default group, if `window` is `NULL` or if `window` does not have an explicit window group.

Parameters

window	a GtkWindow, or `NULL`.	[allow-none]

Returns

the GtkWindowGroup for a window or the default group.

[transfer none]

Since: 2.10

---

gtk_window_has_group ()

```
gboolean
gtk_window_has_group (GtkWindow *window);
```

Returns whether `window` has an explicit window group.

Parameters

window	a GtkWindow	

Returns

`TRUE` if `window` has an explicit window group.

Since 2.22

---

gtk_window_get_window_type ()

```
GtkWindowType
gtk_window_get_window_type (GtkWindow *window);
```

Gets the type of the window. See GtkWindowType.

Parameters

window	a GtkWindow

Returns

the type of the window

Since: 2.20

---

gtk_window_move ()

```
void
gtk_window_move (GtkWindow *window,
                 gint x,
                 gint y);
```

Asks the window manager to move `window` to the given position. Window managers are free to ignore this; most window managers ignore requests for initial window positions (instead using a user-defined placement algorithm) and honor requests after the window has already been shown.

Note: the position is the position of the gravity-determined reference point for the window. The gravity determines two things: first, the location of the reference point in root window coordinates; and second, which point on the window is positioned at the reference point.

By default the gravity is GDK_GRAVITY_NORTH_WEST, so the reference point is simply the `x` , `y` supplied to `gtk_window_move()`. The top-left corner of the window decorations (aka window frame or border) will be placed at `x` , `y` . Therefore, to position a window at the top left of the screen, you want to use the default gravity (which is GDK_GRAVITY_NORTH_WEST) and move the window to 0,0.

To position a window at the bottom right corner of the screen, you would set GDK_GRAVITY_SOUTH_EAST, which means that the reference point is at `x` + the window width and `y` + the window height, and the bottom-right corner of the window border will be placed at that reference point. So, to place a window in the bottom right corner you would first set gravity to south east, then write: `gtk_window_move (window, gdk_screen_width()` - window_width, `gdk_screen_height()` - window_height) (note that this example does not take multi-head scenarios into account).

The Extended Window Manager Hints Specification has a nice table of gravities in the “implementation notes” section.

The `gtk_window_get_position()` documentation may also be relevant.

Parameters

window	a GtkWindow
x	X coordinate to move window to
y	Y coordinate to move window to

---

gtk_window_parse_geometry ()

```
gboolean
gtk_window_parse_geometry (GtkWindow *window,
                           const gchar *geometry);
```

`gtk_window_parse_geometry` has been deprecated since version 3.20 and should not be used in newly-written code.

Geometry handling in GTK is deprecated.

Parses a standard X Window System geometry string - see the manual page for X (type “man X”) for details on this. `gtk_window_parse_geometry()` does work on all GTK+ ports including Win32 but is primarily intended for an X environment.

If either a size or a position can be extracted from the geometry string, `gtk_window_parse_geometry()` returns `TRUE` and calls `gtk_window_set_default_size()` and/or `gtk_window_move()` to resize/move the window.

If `gtk_window_parse_geometry()` returns `TRUE`, it will also set the GDK_HINT_USER_POS and/or GDK_HINT_USER_SIZE hints indicating to the window manager that the size/position of the window was user-specified. This causes most window managers to honor the geometry.

Note that for `gtk_window_parse_geometry()` to work as expected, it has to be called when the window has its “final” size, i.e. after calling `gtk_widget_show_all()` on the contents and `gtk_window_set_geometry_hints()` on the window.

```
#include <gtk/gtk.h>

static void
fill_with_content (GtkWidget *vbox)
{
  // fill with content...
}

int
main (int argc, char *argv[])
{
  GtkWidget *window, *vbox;
  GdkGeometry size_hints = {
    100, 50, 0, 0, 100, 50, 10,
    10, 0.0, 0.0, GDK_GRAVITY_NORTH_WEST
  };

  gtk_init (&argc, &argv);

  window = gtk_window_new (GTK_WINDOW_TOPLEVEL);
  vbox = gtk_box_new (GTK_ORIENTATION_VERTICAL, 0);

  gtk_container_add (GTK_CONTAINER (window), vbox);
  fill_with_content (vbox);
  gtk_widget_show_all (vbox);

  gtk_window_set_geometry_hints (GTK_WINDOW (window),
                             NULL,
                             &size_hints,
                             GDK_HINT_MIN_SIZE |
                             GDK_HINT_BASE_SIZE |
                             GDK_HINT_RESIZE_INC);

  if (argc > 1)
    {
      gboolean res;
      res = gtk_window_parse_geometry (GTK_WINDOW (window),
                                       argv[1]);
      if (! res)
        fprintf (stderr,
                 "Failed to parse “%s”\n",
                 argv[1]);
    }

  gtk_widget_show_all (window);
  gtk_main ();

  return 0;
}
```

Parameters

window	a GtkWindow
geometry	geometry string

Returns

`TRUE` if string was parsed successfully

gtk_window_reshow_with_initial_size ()

```
void
gtk_window_reshow_with_initial_size (GtkWindow *window);
```

`gtk_window_reshow_with_initial_size` has been deprecated since version 3.10 and should not be used in newly-written code.

GUI builders can call `gtk_widget_hide()`, `gtk_widget_unrealize()` and then `gtk_widget_show()` on `window` themselves, if they still need this functionality.

Hides `window` , then reshows it, resetting the default size and position of the window. Used by GUI builders only.

Parameters

window	a GtkWindow

---

gtk_window_resize ()

```
void
gtk_window_resize (GtkWindow *window,
                   gint width,
                   gint height);
```

Resizes the window as if the user had done so, obeying geometry constraints. The default geometry constraint is that windows may not be smaller than their size request; to override this constraint, call `gtk_widget_set_size_request()` to set the window's request to a smaller value.

If `gtk_window_resize()` is called before showing a window for the first time, it overrides any default size set with `gtk_window_set_default_size()`.

Windows may not be resized smaller than 1 by 1 pixels.

When using client side decorations, GTK+ will do its best to adjust the given size so that the resulting window size matches the requested size without the title bar, borders and shadows added for the client side decorations, but there is no guarantee that the result will be totally accurate because these widgets added for client side decorations depend on the theme and may not be realized or visible at the time `gtk_window_resize()` is issued.

If the GtkWindow has a titlebar widget (see `gtk_window_set_titlebar()`), then typically, `gtk_window_resize()` will compensate for the height of the titlebar widget only if the height is known when the resulting GtkWindow configuration is issued. For example, if new widgets are added after the GtkWindow configuration and cause the titlebar widget to grow in height, this will result in a window content smaller that specified by `gtk_window_resize()` and not a larger window.

Parameters

window	a GtkWindow
width	width in pixels to resize the window to
height	height in pixels to resize the window to

---

gtk_window_resize_to_geometry ()

```
void
gtk_window_resize_to_geometry (GtkWindow *window,
                               gint width,
                               gint height);
```

`gtk_window_resize_to_geometry` has been deprecated since version 3.20 and should not be used in newly-written code.

This function does nothing. Use `gtk_window_resize()` and compute the geometry yourself.

Like `gtk_window_resize()`, but `width` and `height` are interpreted in terms of the base size and increment set with gtk_window_set_geometry_hints.

Parameters

window	a GtkWindow
width	width in resize increments to resize the window to
height	height in resize increments to resize the window to

Since: 3.0

---

gtk_window_set_default_icon_list ()

```
void
gtk_window_set_default_icon_list (GList *list);
```

Sets an icon list to be used as fallback for windows that haven't had `gtk_window_set_icon_list()` called on them to set up a window-specific icon list. This function allows you to set up the icon for all windows in your app at once.

See `gtk_window_set_icon_list()` for more details.

Parameters

list	a list of GdkPixbuf.	[element-type GdkPixbuf][transfer container]

---

gtk_window_set_default_icon ()

```
void
gtk_window_set_default_icon (GdkPixbuf *icon);
```

Sets an icon to be used as fallback for windows that haven't had `gtk_window_set_icon()` called on them from a pixbuf.

Parameters

icon	the icon

Since: 2.4

---

gtk_window_set_default_icon_from_file ()

```
gboolean
gtk_window_set_default_icon_from_file (const gchar *filename,
                                       GError **err);
```

Sets an icon to be used as fallback for windows that haven't had `gtk_window_set_icon_list()` called on them from a file on disk. Warns on failure if `err` is `NULL`.

Parameters

filename	location of icon file.	[type filename]
err	location to store error, or `NULL`.	[allow-none]

Returns

`TRUE` if setting the icon succeeded.

Since: 2.2

---

gtk_window_set_default_icon_name ()

```
void
gtk_window_set_default_icon_name (const gchar *name);
```

Sets an icon to be used as fallback for windows that haven't had `gtk_window_set_icon_list()` called on them from a named themed icon, see `gtk_window_set_icon_name()`.

Parameters

name	the name of the themed icon

Since: 2.6

---

gtk_window_set_icon ()

```
void
gtk_window_set_icon (GtkWindow *window,
                     GdkPixbuf *icon);
```

Sets up the icon representing a GtkWindow. This icon is used when the window is minimized (also known as iconified). Some window managers or desktop environments may also place it in the window frame, or display it in other contexts. On others, the icon is not used at all, so your mileage may vary.

The icon should be provided in whatever size it was naturally drawn; that is, don’t scale the image before passing it to GTK+. Scaling is postponed until the last minute, when the desired final size is known, to allow best quality.

If you have your icon hand-drawn in multiple sizes, use `gtk_window_set_icon_list()`. Then the best size will be used.

This function is equivalent to calling `gtk_window_set_icon_list()` with a 1-element list.

See also `gtk_window_set_default_icon_list()` to set the icon for all windows in your application in one go.

Parameters

window	a GtkWindow	
icon	icon image, or `NULL`.	[allow-none]

---

gtk_window_set_icon_list ()

```
void
gtk_window_set_icon_list (GtkWindow *window,
                          GList *list);
```

Sets up the icon representing a GtkWindow. The icon is used when the window is minimized (also known as iconified). Some window managers or desktop environments may also place it in the window frame, or display it in other contexts. On others, the icon is not used at all, so your mileage may vary.

gtk_window_set_icon_list() allows you to pass in the same icon in several hand-drawn sizes. The list should contain the natural sizes your icon is available in; that is, don’t scale the image before passing it to GTK+. Scaling is postponed until the last minute, when the desired final size is known, to allow best quality.

By passing several sizes, you may improve the final image quality of the icon, by reducing or eliminating automatic image scaling.

Recommended sizes to provide: 16x16, 32x32, 48x48 at minimum, and larger images (64x64, 128x128) if you have them.

See also `gtk_window_set_default_icon_list()` to set the icon for all windows in your application in one go.

Note that transient windows (those who have been set transient for another window using `gtk_window_set_transient_for()`) will inherit their icon from their transient parent. So there’s no need to explicitly set the icon on transient windows.

Parameters

window	a GtkWindow	
list	list of GdkPixbuf.	[element-type GdkPixbuf]

---

gtk_window_set_icon_from_file ()

```
gboolean
gtk_window_set_icon_from_file (GtkWindow *window,
                               const gchar *filename,
                               GError **err);
```

Sets the icon for `window` . Warns on failure if `err` is `NULL`.

This function is equivalent to calling `gtk_window_set_icon()` with a pixbuf created by loading the image from `filename` .

Parameters

window	a GtkWindow	
filename	location of icon file.	[type filename]
err	location to store error, or `NULL`.	[allow-none]

Returns

`TRUE` if setting the icon succeeded.

Since: 2.2

---

gtk_window_set_icon_name ()

```
void
gtk_window_set_icon_name (GtkWindow *window,
                          const gchar *name);
```

Sets the icon for the window from a named themed icon. See the docs for GtkIconTheme for more details. On some platforms, the window icon is not used at all.

Note that this has nothing to do with the WM_ICON_NAME property which is mentioned in the ICCCM.

Parameters

window	a GtkWindow	
name	the name of the themed icon.	[allow-none]

Since: 2.6

---

gtk_window_set_auto_startup_notification ()

```
void
gtk_window_set_auto_startup_notification
                               (gboolean setting);
```

By default, after showing the first GtkWindow, GTK+ calls `gdk_notify_startup_complete()`. Call this function to disable the automatic startup notification. You might do this if your first window is a splash screen, and you want to delay notification until after your real main window has been shown, for example.

In that example, you would disable startup notification temporarily, show your splash screen, then re-enable it so that showing the main window would automatically result in notification.

Parameters

setting	`TRUE` to automatically do startup notification

Since: 2.2

---

gtk_window_get_opacity ()

```
gdouble
gtk_window_get_opacity (GtkWindow *window);
```

`gtk_window_get_opacity` has been deprecated since version 3.8 and should not be used in newly-written code.

Use gtk_widget_get_opacity instead.

Fetches the requested opacity for this window. See `gtk_window_set_opacity()`.

Parameters

window	a GtkWindow

Returns

the requested opacity for this window.

Since: 2.12

---

gtk_window_set_opacity ()

```
void
gtk_window_set_opacity (GtkWindow *window,
                        gdouble opacity);
```

`gtk_window_set_opacity` has been deprecated since version 3.8 and should not be used in newly-written code.

Use gtk_widget_set_opacity instead.

Request the windowing system to make `window` partially transparent, with opacity 0 being fully transparent and 1 fully opaque. (Values of the opacity parameter are clamped to the [0,1] range.) On X11 this has any effect only on X screens with a compositing manager running. See `gtk_widget_is_composited()`. On Windows it should work always.

Note that setting a window's opacity after the window has been shown causes it to flicker once on Windows.

Parameters

window	a GtkWindow
opacity	desired opacity, between 0 and 1

Since: 2.12

---

gtk_window_get_mnemonics_visible ()

```
gboolean
gtk_window_get_mnemonics_visible (GtkWindow *window);
```

Gets the value of the "mnemonics-visible" property.

Parameters

window	a GtkWindow

Returns

`TRUE` if mnemonics are supposed to be visible in this window.

Since: 2.20

---

gtk_window_set_mnemonics_visible ()

```
void
gtk_window_set_mnemonics_visible (GtkWindow *window,
                                  gboolean setting);
```

Sets the “mnemonics-visible” property.

Parameters

window	a GtkWindow
setting	the new value

Since: 2.20

gtk_window_get_focus_visible ()

```
gboolean
gtk_window_get_focus_visible (GtkWindow *window);
```

Gets the value of the “focus-visible” property.

Parameters

window	a GtkWindow

Returns

TRUE if “focus rectangles” are supposed to be visible in this window.

Since: 3.2

---

gtk_window_set_focus_visible ()

```
void
gtk_window_set_focus_visible (GtkWindow *window,
                              gboolean setting);
```

Sets the “focus-visible” property.

Parameters

window	a GtkWindow
setting	the new value

Since: 3.2

---

gtk_window_set_has_resize_grip ()

```
void
gtk_window_set_has_resize_grip (GtkWindow *window,
                                gboolean value);
```

gtk_window_set_has_resize_grip has been deprecated since version 3.14 and should not be used in newly-written code.

Resize grips have been removed.

Sets whether window has a corner resize grip.

Note that the resize grip is only shown if the window is actually resizable and not maximized. Use gtk_window_resize_grip_is_visible() to find out if the resize grip is currently shown.

Parameters

window	a GtkWindow
value	TRUE to allow a resize grip

Since: 3.0

---

gtk_window_get_has_resize_grip ()

```
gboolean
gtk_window_get_has_resize_grip (GtkWindow *window);
```

`gtk_window_get_has_resize_grip` has been deprecated since version 3.14 and should not be used in newly-written code.

Resize grips have been removed.

Determines whether the window may have a resize grip.

Parameters

window	a GtkWindow

Returns

`TRUE` if the window has a resize grip

Since: 3.0

---

gtk_window_resize_grip_is_visible ()

```
gboolean
gtk_window_resize_grip_is_visible (GtkWindow *window);
```

`gtk_window_resize_grip_is_visible` has been deprecated since version 3.14 and should not be used in newly-written code.

Resize grips have been removed.

Determines whether a resize grip is visible for the specified window.

Parameters

window	a GtkWindow

Returns

`TRUE` if a resize grip exists and is visible

Since: 3.0

---

gtk_window_get_resize_grip_area ()

```
gboolean
gtk_window_get_resize_grip_area (GtkWindow *window,
                                 GdkRectangle *rect);
```

`gtk_window_get_resize_grip_area` has been deprecated since version 3.14 and should not be used in newly-written code.

Resize grips have been removed.

If a window has a resize grip, this will retrieve the grip position, width and height into the specified GdkRectangle.

Parameters

window	a GtkWindow	
rect	a pointer to a GdkRectangle which we should store the resize grip area.	[out]

Returns

`TRUE` if the resize grip's area was retrieved

Since: 3.0

---

gtk_window_get_application ()

```
GtkApplication *
gtk_window_get_application (GtkWindow *window);
```

Gets the GtkApplication associated with the window (if any).

Parameters

window	a GtkWindow

Returns

a GtkApplication, or `NULL`.

[nullable][transfer none]

Since: 3.0

---

gtk_window_set_application ()

```
void
gtk_window_set_application (GtkWindow *window,
                            GtkApplication *application);
```

Sets or unsets the GtkApplication associated with the window.

The application will be kept alive for at least as long as it has any windows associated with it (see `g_application_hold()` for a way to keep it alive without windows).

Normally, the connection between the application and the window will remain until the window is destroyed, but you can explicitly remove it by setting the `application` to `NULL`.

This is equivalent to calling `gtk_application_remove_window()` and/or `gtk_application_add_window()` on the old/new applications as relevant.

Parameters

window	a GtkWindow
application	a GtkApplication, or `NULL` to unset. [allow-none]

Since: 3.0

---

gtk_window_set_has_user_ref_count ()

```
void
gtk_window_set_has_user_ref_count (GtkWindow *window,
                                   gboolean setting);
```

Tells GTK+ whether to drop its extra reference to the window when `gtk_widget_destroy()` is called.

This function is only exported for the benefit of language bindings which may need to keep the window alive until their wrapper object is garbage collected. There is no justification for ever calling this function in an application.

Parameters

window	a GtkWindow
setting	the new value

Since: 3.0

---

gtk_window_set_titlebar ()

```
void
gtk_window_set_titlebar (GtkWindow *window,
                         GtkWidget *titlebar);
```

Sets a custom titlebar for `window`.

A typical widget used here is GtkHeaderBar, as it provides various features expected of a titlebar while allowing the addition of child widgets to it.

If you set a custom titlebar, GTK+ will do its best to convince the window manager not to put its own titlebar on the window. Depending on the system, this function may not work for a window that is already visible, so you set the titlebar before calling `gtk_widget_show()`.

Parameters

window	a GtkWindow	
titlebar	the widget to use as titlebar.	[allow-none]

Since: 3.10

---

gtk_window_get_titlebar ()

```
GtkWidget *
gtk_window_get_titlebar (GtkWindow *window);
```

Returns the custom titlebar that has been set with `gtk_window_set_titlebar()`.

Parameters

window	a GtkWindow	

Returns

the custom titlebar, or `NULL`.

[nullable][transfer none]

Since: 3.16

---

gtk_window_set_interactive_debugging ()

```
void
gtk_window_set_interactive_debugging (gboolean enable);
```

Opens or closes the interactive debugger, which offers access to the widget hierarchy of the application and to useful debugging tools.

Parameters

enable	`TRUE` to enable interactive debugging

Since: 3.14

Types and Values

GtkWindow

```
typedef struct _GtkWindow GtkWindow;
```

---

struct GtkWindowClass

```
struct GtkWindowClass {
  GtkBinClass parent_class;


  void     (* set_focus)   (GtkWindow *window,
                             GtkWidget *focus);

  /* G_SIGNAL_ACTION signals for keybindings */

  void     (* activate_focus)   (GtkWindow *window);
  void     (* activate_default) (GtkWindow *window);
  void         (* keys_changed)     (GtkWindow *window);
  gboolean (* enable_debugging) (GtkWindow *window,
                                 gboolean   toggle);
};
```

Members

`set_focus` ()	Sets child as the focus widget for the window.
`activate_focus` ()	Activates the current focused widget within the window.
`activate_default` ()	Activates the default widget for the window.
`keys_changed` ()	Signal gets emitted when the set of accelerators or mnemonics that are associated with window changes.
`enable_debugging` ()	Class handler for the "enable-debugging" keybinding signal. Since: 3.14

---

enum GtkWindowType

A GtkWindow can be one of these types. Most things you'd consider a "window" should have type GTK_WINDOW_TOPLEVEL; windows with this type are managed by the window manager and have a frame by default (call `gtk_window_set_decorated()` to toggle the frame). Windows with type GTK_WINDOW_POPUP are ignored by the window manager; window manager keybindings won't work on them, the window manager won't decorate the window with a frame, many GTK+ features that rely on the window manager will not work (e.g. resize grips and maximization/minimization). GTK_WINDOW_POPUP is used to implement widgets such as GtkMenu or tooltips that you normally don't think of as windows per se.

Nearly all windows should be GTK_WINDOW_TOPLEVEL. In particular, do not use GTK_WINDOW_POPUP just to turn off the window borders; use `gtk_window_set_decorated()` for that.

Members

GTK_WINDOW_TOPLEVEL	A regular window, such as a dialog.
GTK_WINDOW_POPUP	A special window such as a tooltip.

---

enum GtkWindowPosition

Window placement can be influenced using this enumeration. Note that using GTK_WIN_POS_CENTER_ALWAYS is almost always a bad idea. It won't necessarily work well with all window managers or on all windowing systems.

Members

GTK_WIN_POS_NONE	No influence is made on placement.
GTK_WIN_POS_CENTER	Windows should be placed in the center of the screen.
GTK_WIN_POS_MOUSE	Windows should be placed at the current mouse position.
GTK_WIN_POS_CENTER_ALWAYS	Keep window centered as it changes size, etc.
GTK_WIN_POS_CENTER_ON_PARENT	Center the window on its transient parent (see `gtk_window_set_transient_for()`).

Property Details

The `"accept-focus"` property

`"accept-focus"` `gboolean`

Whether the window should receive the input focus.

Flags: Read / Write

Default value: TRUE

Since: 2.4

The `"application"` property

`"application"` `GtkApplication *`

The GtkApplication associated with the window.

The application will be kept alive for at least as long as it has any windows associated with it (see `g_application_hold()` for a way to keep it alive without windows).

Normally, the connection between the application and the window will remain until the window is destroyed, but you can explicitly remove it by setting the :application property to `NULL`.

Flags: Read / Write

Since: 3.0

The “attached-to” property

“attached-to” GtkWidget *

The widget to which this window is attached. See gtk_window_set_attached_to().

Examples of places where specifying this relation is useful are for instance a GtkMenu created by a GtkComboBox, a completion popup window created by GtkEntry or a typeahead search entry created by GtkTreeView.

Flags: Read / Write / Construct

Since: 3.4

The “decorated” property

“decorated” gboolean

Whether the window should be decorated by the window manager.

Flags: Read / Write

Default value: TRUE

Since: 2.4

The “default-height” property

“default-height” gint

The default height of the window, used when initially showing the window.

Flags: Read / Write

Allowed values: >= -1

Default value: -1

The “default-width” property

“default-width” gint

The default width of the window, used when initially showing the window.

Flags: Read / Write

Allowed values: >= -1

Default value: -1

The “deletable” property

“deletable” gboolean

Whether the window frame should have a close button.

Flags: Read / Write

Default value: TRUE

Since: 2.10

The “destroy-with-parent” property

“destroy-with-parent” gboolean

If this window should be destroyed when the parent is destroyed.

Flags: Read / Write

Default value: FALSE

The “focus-on-map” property

“focus-on-map” gboolean

Whether the window should receive the input focus when mapped.

Flags: Read / Write

Default value: TRUE

Since: 2.6

The “focus-visible” property

“focus-visible” gboolean

Whether 'focus rectangles' are currently visible in this window.

This property is maintained by GTK+ based on user input and should not be set by applications.

Flags: Read / Write

Default value: TRUE

Since: 2.20

The “gravity” property

“gravity” GdkGravity

The window gravity of the window. See gtk_window_move() and GdkGravity for more details about window gravity.

Flags: Read / Write

Default value: GDK_GRAVITY_NORTH_WEST

Since: 2.4

The “has-resize-grip” property

“has-resize-grip” gboolean

Whether the window has a corner resize grip.

Note that the resize grip is only shown if the window is actually resizable and not maximized. Use “resize-grip-visible” to find out if the resize grip is currently shown.

GtkWindow:has-resize-grip has been deprecated since version 3.14 and should not be used in newly-written code.

Resize grips have been removed.

Flags: Read / Write

Default value: FALSE

Since: 3.0

The “has-toplevel-focus” property

“has-toplevel-focus” gboolean

Whether the input focus is within this GtkWindow.

Flags: Read

Default value: FALSE

The “hide-titlebar-when-maximized” property

“hide-titlebar-when-maximized” gboolean

Whether the titlebar should be hidden during maximization.

Flags: Read / Write

Default value: FALSE

Since: 3.4

The “icon” property

“icon” GdkPixbuf *

Icon for this window.

Flags: Read / Write

The “icon-name” property

“icon-name” gchar *

The :icon-name property specifies the name of the themed icon to use as the window icon. See GtkIconTheme for more details.

Flags: Read / Write

Default value: NULL

Since: 2.6

The “is-active” property

“is-active” gboolean

Whether the toplevel is the current active window.

Flags: Read

Default value: FALSE

The “is-maximized” property

“is-maximized” gboolean

Whether the window is maximized.

Flags: Read

Default value: FALSE

The “mnemonics-visible” property

“mnemonics-visible” gboolean

Whether mnemonics are currently visible in this window.

This property is maintained by GTK+ based on user input, and should not be set by applications.

Flags: Read / Write

Default value: TRUE

Since: 2.20

The “modal” property

“modal” gboolean

If TRUE, the window is modal (other windows are not usable while this one is up).

Flags: Read / Write

Default value: FALSE

The “resizable” property

“resizable” gboolean

If TRUE, users can resize the window.

Flags: Read / Write

Default value: TRUE

The “resize-grip-visible” property

“resize-grip-visible” gboolean

Whether a corner resize grip is currently shown.

GtkWindow:resize-grip-visible has been deprecated since version 3.14 and should not be used in newly-written code.

Resize grips have been removed.

Flags: Read

Default value: FALSE

Since: 3.0

The “role” property

“role” gchar *

Unique identifier for the window to be used when restoring a session.

Flags: Read / Write

Default value: NULL

The “screen” property

“screen” GdkScreen *

The screen where this window will be displayed.

Flags: Read / Write

The “skip-pager-hint” property

“skip-pager-hint” gboolean

TRUE if the window should not be in the pager.

Flags: Read / Write

Default value: FALSE

The “skip-taskbar-hint” property

“skip-taskbar-hint” gboolean

TRUE if the window should not be in the task bar.

Flags: Read / Write

Default value: FALSE

The “startup-id” property

“startup-id” gchar *

The :startup-id is a write-only property for setting window's startup notification identifier. See gtk_window_set_startup_id() for more details.

Flags: Write

Default value: NULL

Since: 2.12

The “title” property

“title” gchar *

The title of the window.

Flags: Read / Write

Default value: NULL

The “transient-for” property

“transient-for” GtkWindow *

The transient parent of the window. See gtk_window_set_transient_for() for more details about transient windows.

Flags: Read / Write / Construct

Since: 2.10

The “type” property

“type” GtkWindowType

The type of the window.

Flags: Read / Write / Construct Only

Default value: GTK_WINDOW_TOPLEVEL

The “type-hint” property

“type-hint” GdkWindowTypeHint

Hint to help the desktop environment understand what kind of window this is and how to treat it.

Flags: Read / Write

Default value: GDK_WINDOW_TYPE_HINT_NORMAL

The “urgency-hint” property

“urgency-hint” gboolean

TRUE if the window should be brought to the user's attention.

Flags: Read / Write

Default value: FALSE

The “window-position” property

“window-position” GtkWindowPosition

The initial position of the window.

Flags: Read / Write

Default value: GTK_WIN_POS_NONE

Style Property Details

The `“decoration-button-layout”` style property

```
“decoration-button-layout” gchar *
```

Decorated button layout.

Flags: Read

Default value: "menu:close"

The `“decoration-resize-handle”` style property

```
“decoration-resize-handle” gint
```

Decoration resize handle size.

Flags: Read / Write

Allowed values: >= 0

Default value: 20

Signal Details

The `"activate-default"` signal

```
void
user_function (GtkWindow *window,
               gpointer   user_data)
```

The ::activate-default signal is a keybinding signal which gets emitted when the user activates the default widget of `window`.

Parameters

window	the window which received the signal
user_data	user data set when the signal handler was connected.

Flags: Action

The `"activate-focus"` signal

```
void
user_function (GtkWindow *window,
               gpointer   user_data)
```

The ::activate-focus signal is a keybinding signal which gets emitted when the user activates the currently focused widget of `window`.

Parameters

window	the window which received the signal
user_data	user data set when the signal handler was connected.

Flags: Action

The “enable-debugging” signal

```
gboolean
user_function (GtkWindow *window,
               gboolean   toggle,
               gpointer   user_data)
```

The ::enable-debugging signal is a keybinding signal which gets emitted when the user enables or disables interactive debugging. When `toggle` is `TRUE`, interactive debugging is toggled on or off, when it is `FALSE`, the debugger will be pointed at the widget under the pointer.

The default bindings for this signal are Ctrl-Shift-I and Ctrl-Shift-D.

Return: `TRUE` if the key binding was handled

Parameters

window	the window on which the signal is emitted
toggle	toggle the debugger
user_data	user data set when the signal handler was connected.

Flags: Action

The “keys-changed” signal

```
void
user_function (GtkWindow *window,
               gpointer   user_data)
```

The ::keys-changed signal gets emitted when the set of accelerators or mnemonics that are associated with `window` changes.

Parameters

window	the window which received the signal
user_data	user data set when the signal handler was connected.

Flags: Run First

The “set-focus” signal

```
void
user_function (GtkWindow *window,
               GtkWidget *widget,
               gpointer   user_data)
```

Flags: Run Last

GtkDialog

GtkDialog — Create popup windows

Functions

GtkWidget *	gtk_dialog_new ()
GtkWidget *	gtk_dialog_new_with_buttons ()
gint	gtk_dialog_run ()
void	gtk_dialog_response ()
GtkWidget *	gtk_dialog_add_button ()
void	gtk_dialog_add_buttons ()
void	gtk_dialog_add_action_widget ()
void	gtk_dialog_set_default_response ()
void	gtk_dialog_set_response_sensitive ()
gint	gtk_dialog_get_response_for_widget ()
GtkWidget *	gtk_dialog_get_widget_for_response ()
GtkWidget *	gtk_dialog_get_action_area ()
GtkWidget *	gtk_dialog_get_content_area ()
GtkWidget *	gtk_dialog_get_header_bar ()
gboolean	gtk_alternative_dialog_button_order ()
void	gtk_dialog_set_alternative_button_order ()
void	gtk_dialog_set_alternative_button_order_from_array ()

Properties

gint	use-header-bar	Read / Write / Construct Only

Style Properties

gint	action-area-border	Read
gint	button-spacing	Read
gint	content-area-border	Read
gint	content-area-spacing	Read

Signals

void	close	Action
void	response	Run Last

Types and Values

struct	GtkDialog
struct	GtkDialogClass
enum	GtkDialogFlags
enum	GtkResponseType

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkContainer
            └── GtkBin
                └── GtkWindow
                    └── GtkDialog
                        ├── GtkAboutDialog
                        ├── GtkAppChooserDialog
                        ├── GtkColorChooserDialog
                        ├── GtkColorSelectionDialog
                        ├── GtkFileChooserDialog
                        ├── GtkFontChooserDialog
                        ├── GtkFontSelectionDialog
                        ├── GtkMessageDialog
                        ├── GtkPageSetupUnixDialog
                        ├── GtkPrintUnixDialog
                        └── GtkRecentChooserDialog
```

Implemented Interfaces

GtkDialog implements AtkImplementorIface and GtkBuildable.

Includes

```
#include <gtk/gtk.h>
```

Description

Dialog boxes are a convenient way to prompt the user for a small amount of input, e.g. to display a message, ask a question, or anything else that does not require extensive effort on the user's part.

GTK+ treats a dialog as a window split vertically. The top section is a GtkVBox, and is where widgets such as a GtkLabel or a GtkEntry should be packed. The bottom area is known as the "action area". This is generally used for packing buttons into the dialog which may perform functions such as cancel, ok, or apply.

GtkDialog boxes are created with a call to `gtk_dialog_new()` or `gtk_dialog_new_with_buttons()`. `gtk_dialog_new_with_buttons()` is recommended; it allows you to set the dialog title, some convenient flags, and add simple buttons.

If "dialog" is a newly created dialog, the two primary areas of the window can be accessed through `gtk_dialog_get_content_area()` and `gtk_dialog_get_action_area()`, as can be seen from the example below.

A "modal" dialog (that is, one which freezes the rest of the application from user input), can be created by calling `gtk_window_set_modal()` on the dialog. Use the `GTK_WINDOW()` macro to cast the widget returned from `gtk_dialog_new()` into a GtkWindow. When using `gtk_dialog_new_with_buttons()` you can also pass the GTK_DIALOG_MODAL flag to make a dialog modal.

If you add buttons to GtkDialog using `gtk_dialog_new_with_buttons()`, `gtk_dialog_add_button()`, `gtk_dialog_add_buttons()`, or `gtk_dialog_add_action_widget()`, clicking the button will emit a signal called "response" with a response ID that you specified. GTK+ will never assign a meaning to positive response IDs; these are entirely user-defined. But for convenience, you can use the response IDs in the GtkResponseType enumeration (these all have values less than zero). If a dialog receives a delete event, the "response" signal will be emitted with a response ID of GTK_RESPONSE_DELETE_EVENT.

If you want to block waiting for a dialog to return before returning control flow to your code, you can call

gtk_dialog_run(). This function enters a recursive main loop and waits for the user to respond to the dialog, returning the response ID corresponding to the button the user clicked.

For the simple dialog in the following example, in reality you'd probably use GtkMessageDialog to save yourself some effort. But you'd need to create the dialog contents manually if you had more than a simple message in the dialog.

An example for simple GtkDialog usage:

```
1   // Function to open a dialog box with a message
2   void
3   quick_message (GtkWindow *parent, gchar *message)
4   {
5    GtkWidget *dialog, *label, *content_area;
6    GtkDialogFlags flags;
7
8    // Create the widgets
9    flags = GTK_DIALOG_DESTROY_WITH_PARENT;
10   dialog = gtk_dialog_new_with_buttons ("Message",
11                                         parent,
12                                         flags,
13                                         _("_OK"),
14                                         GTK_RESPONSE_NONE,
15                                         NULL);
16   content_area = gtk_dialog_get_content_area (GTK_DIALOG (dialog));
17   label = gtk_label_new (message);
18
19   // Ensure that the dialog box is destroyed when the user responds
20
21   g_signal_connect_swapped (dialog,
22                             "response",
23                             G_CALLBACK (gtk_widget_destroy),
24                             dialog);
25
26   // Add the label, and show everything we've added
27
28   gtk_container_add (GTK_CONTAINER (content_area), label);
29   gtk_widget_show_all (dialog);
30  }
```

GtkDialog as GtkBuildable

The GtkDialog implementation of the GtkBuildable interface exposes the `vbox` and `action_area` as internal children with the names “vbox” and “action_area”.

GtkDialog supports a custom <action-widgets> element, which can contain multiple <action-widget> elements. The “response” attribute specifies a numeric response, and the content of the element is the id of widget (which should be a child of the dialogs `action_area`). To mark a response as default, set the “default“ attribute of the <action-widget> element to true.

GtkDialog supports adding action widgets by specifying “action“ as the “type“ attribute of a <child> element. The widget will be added either to the action area or the headerbar of the dialog, depending on the “use-header-bar“ property. The response id has to be associated with the action widget using the <action-widgets> element.

An example of a GtkDialog UI definition fragment:

```
1   <object class="GtkDialog" id="dialog1">
2     <child type="action">
3       <object class="GtkButton" id="button_cancel"/>
4     </child>
5     <child type="action">
6       <object class="GtkButton" id="button_ok">
7         <property name="can-default">True</property>
8       </object>
9     </child>
10    <action-widgets>
11      <action-widget response="cancel">button_cancel</action-widget>
12      <action-widget response="ok" default="true">button_ok</action-widget>
13    </action-widgets>
14  </object>
```

Functions

gtk_dialog_new ()

```
GtkWidget *
gtk_dialog_new (void);
```

Creates a new dialog box.

Widgets should not be packed into this GtkWindow directly, but into the `vbox` and `action_area` , as described above.

Returns

the new dialog as a GtkWidget

gtk_dialog_new_with_buttons ()

```
GtkWidget *
gtk_dialog_new_with_buttons (const gchar *title,
                             GtkWindow *parent,
                             GtkDialogFlags flags,
                             const gchar *first_button_text,
                             ...);
```

Creates a new GtkDialog with title `title` (or `NULL` for the default title; see `gtk_window_set_title()`) and transient parent `parent` (or `NULL` for none; see `gtk_window_set_transient_for()`). The `flags` argument can be used to make the dialog modal (GTK_DIALOG_MODAL) and/or to have it destroyed along with its transient parent (GTK_DIALOG_DESTROY_WITH_PARENT). After `flags` , button text/response ID pairs should be listed, with a `NULL` pointer ending the list. Button text can be arbitrary text. A response ID can be any positive number, or one of the values in the GtkResponseType enumeration. If the user clicks one of these dialog buttons, GtkDialog will emit the "response" signal with the corresponding response ID. If a GtkDialog receives the "delete-event" signal, it will emit ::response with a response ID of GTK_RESPONSE_DELETE_EVENT. However, destroying a dialog does not emit the ::response signal; so be careful relying on ::response when using the GTK_DIALOG_DESTROY_WITH_PARENT flag. Buttons are from left to right, so the first button in the list will be the leftmost button in the dialog.

Here's a simple example:

```
1  GtkWidget *main_app_window; // Window the dialog should show up on
2  GtkWidget *dialog;
3  GtkDialogFlags flags = GTK_DIALOG_MODAL | GTK_DIALOG_DESTROY_WITH_PARENT;
4  dialog = gtk_dialog_new_with_buttons ("My dialog",
5                                        main_app_window,
6                                        flags,
7                                        _("_OK"),
8                                        GTK_RESPONSE_ACCEPT,
9                                        _("_Cancel"),
10                                       GTK_RESPONSE_REJECT,
11                                       NULL);
```

Parameters

title	Title of the dialog, or `NULL`.	[allow-none]
parent	Transient parent of the dialog, or `NULL`.	[allow-none]
flags	from GtkDialogFlags	
first_button_text	text to go in first button, or `NULL`.	[allow-none]
...	response ID for first button, then additional buttons, ending with `NULL`	

Returns

a new GtkDialog

---

gtk_dialog_run ()

```
gint
gtk_dialog_run (GtkDialog *dialog);
```

Blocks in a recursive main loop until the `dialog` either emits the "response" signal, or is destroyed. If the dialog is destroyed during the call to `gtk_dialog_run()`, `gtk_dialog_run()` returns GTK_RESPONSE_NONE. Otherwise, it returns the response ID from the ::response signal emission.

Before entering the recursive main loop, `gtk_dialog_run()` calls `gtk_widget_show()` on the dialog for you. Note that you still need to show any children of the dialog yourself.

During `gtk_dialog_run()`, the default behavior of "delete-event" is disabled; if the dialog receives ::delete_event, it will not be destroyed as windows usually are, and `gtk_dialog_run()` will return GTK_RESPONSE_DELETE_EVENT. Also, during `gtk_dialog_run()` the dialog will be modal. You can force `gtk_dialog_run()` to return at any time by calling `gtk_dialog_response()` to emit the ::response signal. Destroying the dialog during `gtk_dialog_run()` is a very bad idea, because your post-run code won't know whether the dialog was destroyed or not.

After `gtk_dialog_run()` returns, you are responsible for hiding or destroying the dialog if you wish to do so.

Typical usage of this function might be:

```
1  GtkWidget *dialog = gtk_dialog_new ();
2  // Set up dialog...
3
4  int result = gtk_dialog_run (GTK_DIALOG (dialog));
5  switch (result)
6    {
7      case GTK_RESPONSE_ACCEPT:
8         // do_application_specific_something ();
9         break;
10     default:
11        // do_nothing_since_dialog_was_cancelled ();
12        break;
13   }
14 gtk_widget_destroy (dialog);
```

Note that even though the recursive main loop gives the effect of a modal dialog (it prevents the user from interacting with other windows in the same window group while the dialog is run), callbacks such as timeouts, IO channel watches, DND drops, etc, will be triggered during a `gtk_dialog_run()` call.

Parameters

dialog	a GtkDialog

Returns

response ID

gtk_dialog_response ()

```
void
gtk_dialog_response (GtkDialog *dialog,
                     gint response_id);
```

Emits the “response” signal with the given response ID. Used to indicate that the user has responded to the dialog in some way; typically either you or `gtk_dialog_run()` will be monitoring the ::response signal and take appropriate action.

Parameters

dialog	a GtkDialog
response_id	response ID

---

gtk_dialog_add_button ()

```
GtkWidget *
gtk_dialog_add_button (GtkDialog *dialog,
                       const gchar *button_text,
                       gint response_id);
```

Adds a button with the given text and sets things up so that clicking the button will emit the “response” signal with the given `response_id` . The button is appended to the end of the dialog’s action area. The button widget is returned, but usually you don’t need it.

Parameters

dialog	a GtkDialog
button_text	text of button
response_id	response ID for the button

Returns

the GtkButton widget that was added.

[transfer none]

---

gtk_dialog_add_buttons ()

```
void
gtk_dialog_add_buttons (GtkDialog *dialog,
                        const gchar *first_button_text,
                        ...);
```

Adds more buttons, same as calling `gtk_dialog_add_button()` repeatedly. The variable argument list should be `NULL`-terminated as with `gtk_dialog_new_with_buttons()`. Each button must have both text and response ID.

Parameters

dialog	a GtkDialog
first_button_text	button text
...	response ID for first button, then more text-response_id pairs

---

gtk_dialog_add_action_widget ()

```
void
gtk_dialog_add_action_widget (GtkDialog *dialog,
                              GtkWidget *child,
                              gint response_id);
```

Adds an activatable widget to the action area of a GtkDialog, connecting a signal handler that will emit the "response" signal on the dialog when the widget is activated. The widget is appended to the end of the dialog's action area. If you want to add a non-activatable widget, simply pack it into the `action_area` field of the GtkDialog struct.

Parameters

dialog	a GtkDialog
child	an activatable widget
response_id	response ID for `child`

---

gtk_dialog_set_default_response ()

```
void
gtk_dialog_set_default_response (GtkDialog *dialog,
                                 gint response_id);
```

Sets the last widget in the dialog's action area with the given `response_id` as the default widget for the dialog. Pressing "Enter" normally activates the default widget.

Parameters

dialog	a GtkDialog
response_id	a response ID

---

gtk_dialog_set_response_sensitive ()

```
void
gtk_dialog_set_response_sensitive (GtkDialog *dialog,
                                   gint response_id,
                                   gboolean setting);
```

Calls `gtk_widget_set_sensitive (widget, @setting)` for each widget in the dialog's action area with the given `response_id` . A convenient way to sensitize/desensitize dialog buttons.

Parameters

dialog	a GtkDialog
response_id	a response ID
setting	`TRUE` for sensitive

---

gtk_dialog_get_response_for_widget ()

```
gint
gtk_dialog_get_response_for_widget (GtkDialog *dialog,
                                    GtkWidget *widget);
```

Gets the response id of a widget in the action area of a dialog.

Parameters

dialog	a GtkDialog
widget	a widget in the action area of `dialog`

Returns

the response id of `widget` , or `GTK_RESPONSE_NONE` if `widget` doesn't have a response id set.

Since: 2.8

---

gtk_dialog_get_widget_for_response ()

```
GtkWidget *
gtk_dialog_get_widget_for_response (GtkDialog *dialog,
                                    gint response_id);
```

Gets the widget button that uses the given response ID in the action area of a dialog.

Parameters

dialog	a GtkDialog
response_id	the response ID used by the `dialog` widget

Returns

the `widget` button that uses the given `response_id` , or `NULL`.

[nullable][transfer none]

Since: 2.20

---

gtk_dialog_get_action_area ()

```
GtkWidget *
gtk_dialog_get_action_area (GtkDialog *dialog);
```

`gtk_dialog_get_action_area` has been deprecated since version 3.12 and should not be used in newly-written code.

Direct access to the action area is discouraged; use `gtk_dialog_add_button()`, etc.

Returns the action area of `dialog` .

Parameters

dialog	a GtkDialog

Returns

the action area.

[transfer none]

Since: 2.14

---

gtk_dialog_get_content_area ()

```
GtkWidget *
gtk_dialog_get_content_area (GtkDialog *dialog);
```

Returns the content area of `dialog` .

Parameters

dialog	a GtkDialog

Returns

the content area GtkBox.

[type Gtk.Box][transfer none]

Since: 2.14

---

gtk_dialog_get_header_bar ()

```
GtkWidget *
gtk_dialog_get_header_bar (GtkDialog *dialog);
```

Returns the header bar of `dialog` . Note that the headerbar is only used by the dialog if the "use-header-bar" property is `TRUE`.

Parameters

dialog	a GtkDialog	

Returns

the header bar.

[transfer none]

Since: 3.12

---

gtk_alternative_dialog_button_order ()

```
gboolean
gtk_alternative_dialog_button_order (GdkScreen *screen);
```

`gtk_alternative_dialog_button_order` has been deprecated since version 3.10 and should not be used in newly-written code.

Deprecated

Returns `TRUE` if dialogs are expected to use an alternative button order on the screen `screen` . See `gtk_dialog_set_alternative_button_order()` for more details about alternative button order.

If you need to use this function, you should probably connect to the ::notify:gtk-alternative-button-order signal on the GtkSettings object associated to `screen` , in order to be notified if the button order setting changes.

Parameters

screen	a GdkScreen, or `NULL` to use the default screen.	[allow-none]

Returns

Whether the alternative button order should be used

Since: 2.6

---

gtk_dialog_set_alternative_button_order ()

```
void
gtk_dialog_set_alternative_button_order
                                (GtkDialog *dialog,
                                 gint first_response_id,
                                 ...);
```

`gtk_dialog_set_alternative_button_order` has been deprecated since version 3.10 and should not be used in newly-written code.

Deprecated

Sets an alternative button order. If the “gtk-alternative-button-order” setting is set to `TRUE`, the dialog buttons are reordered according to the order of the response ids passed to this function.

By default, GTK+ dialogs use the button order advocated by the GNOME Human Interface Guidelines with the affirmative button at the far right, and the cancel button left of it. But the builtin GTK+ dialogs and GtkMessageDialogs do provide an alternative button order, which is more suitable on some platforms, e.g. Windows.

Use this function after adding all the buttons to your dialog, as the following example shows:

```
1   cancel_button = gtk_dialog_add_button (GTK_DIALOG (dialog),
2                                          _("_Cancel"),
3                                          GTK_RESPONSE_CANCEL);
4
5   ok_button = gtk_dialog_add_button (GTK_DIALOG (dialog),
6                                      _("_OK"),
7                                      GTK_RESPONSE_OK);
8
9   gtk_widget_grab_default (ok_button);
10
11  help_button = gtk_dialog_add_button (GTK_DIALOG (dialog),
12                                       _("_Help"),
13                                       GTK_RESPONSE_HELP);
14
15  gtk_dialog_set_alternative_button_order (GTK_DIALOG (dialog),
16                                           GTK_RESPONSE_OK,
17                                           GTK_RESPONSE_CANCEL,
18                                           GTK_RESPONSE_HELP,
19                                           -1);
```

Parameters

dialog	a GtkDialog
first_response_id	a response id used by one `dialog` ’s buttons
...	a list of more response ids of `dialog` ’s buttons, terminated by -1

Since: 2.6

gtk_dialog_set_alternative_button_order_from_array ()

```
void
gtk_dialog_set_alternative_button_order_from_array
                               (GtkDialog *dialog,
                                gint n_params,
                                gint *new_order);
```

`gtk_dialog_set_alternative_button_order_from_array` has been deprecated since version 3.10 and should not be used in newly-written code.

Deprecated

Sets an alternative button order. If the "gtk-alternative-button-order" setting is set to `TRUE`, the dialog buttons are reordered according to the order of the response ids in `new_order` .

See `gtk_dialog_set_alternative_button_order()` for more information.

This function is for use by language bindings.

Parameters

dialog	a GtkDialog	
n_params	the number of response ids in `new_order`	
new_order	an array of response ids of `dialog` 's buttons.	[array length=n_params]

Since: 2.6

Types and Values

struct GtkDialog

```
struct GtkDialog;
```

The GtkDialog contains only private fields and should not be directly accessed.

---

struct GtkDialogClass

```
struct GtkDialogClass {
  GtkWindowClass parent_class;


  void (* response) (GtkDialog *dialog, gint response_id);

  /* Keybinding signals */

  void (* close)    (GtkDialog *dialog);
};
```

Members

`response` ()	Signal emitted when an action widget is activated.
`close` ()	Signal emitted when the user uses a keybinding to close the dialog.

---

enum GtkDialogFlags

Flags used to influence dialog construction.

Members

GTK_DIALOG_MODAL	Make the constructed dialog modal, see `gtk_window_set_modal()`
GTK_DIALOG_DESTROY_WITH_PARENT	Destroy the dialog when its parent is destroyed, see `gtk_window_set_destroy_with_parent()`
GTK_DIALOG_USE_HEADER_BAR	Create dialog with actions in header bar instead of action area. Since 3.12.

---

enum GtkResponseType

Predefined values for use as response ids in `gtk_dialog_add_button()`. All predefined values are negative; GTK+ leaves values of 0 or greater for application-defined response ids.

Members

GTK_RESPONSE_NONE	Returned if an action widget has no response id, or if the dialog gets programmatically hidden or destroyed
GTK_RESPONSE_REJECT	Generic response id, not used by GTK+ dialogs
GTK_RESPONSE_ACCEPT	Generic response id, not used by GTK+ dialogs
GTK_RESPONSE_DELETE_EVENT	Returned if the dialog is deleted
GTK_RESPONSE_OK	Returned by OK buttons in GTK+ dialogs
GTK_RESPONSE_CANCEL	Returned by Cancel buttons in GTK+ dialogs
GTK_RESPONSE_CLOSE	Returned by Close buttons in GTK+ dialogs
GTK_RESPONSE_YES	Returned by Yes buttons in GTK+

	dialogs
GTK_RESPONSE_NO	Returned by No buttons in GTK+ dialogs
GTK_RESPONSE_APPLY	Returned by Apply buttons in GTK+ dialogs
GTK_RESPONSE_HELP	Returned by Help buttons in GTK+ dialogs

Property Details

The `"use-header-bar"` property

`"use-header-bar"` `gint`

`TRUE` if the dialog uses a GtkHeaderBar for action buttons instead of the action-area.

For technical reasons, this property is declared as an integer property, but you should only set it to `TRUE` or `FALSE`.

Flags: Read / Write / Construct Only

Allowed values: [-1,1]

Default value: -1

Since: 3.12

Style Property Details

The `"action-area-border"` style property

`"action-area-border"` `gint`

The default border width used around the action area of the dialog, as returned by `gtk_dialog_get_action_area()`, unless `gtk_container_set_border_width()` was called on that widget directly.

Flags: Read

Allowed values: >= 0

Default value: 5

The `"button-spacing"` style property

`"button-spacing"` `gint`

Spacing between buttons.

Flags: Read

Allowed values: >= 0

Default value: 6

The “content-area-border” style property

“content-area-border” gint

The default border width used around the content area of the dialog, as returned by gtk_dialog_get_content_area(), unless gtk_container_set_border_width() was called on that widget directly.

Flags: Read

Allowed values: >= 0

Default value: 2

The “content-area-spacing” style property

“content-area-spacing” gint

The default spacing used between elements of the content area of the dialog, as returned by gtk_dialog_get_content_area(), unless gtk_box_set_spacing() was called on that widget directly.

Flags: Read

Allowed values: >= 0

Default value: 0

Since: 2.16

Signal Details

The “close” signal

```
void
user_function (GtkDialog *dialog,
               gpointer   user_data)
```

The ::close signal is a keybinding signal which gets emitted when the user uses a keybinding to close the dialog.

The default binding for this signal is the Escape key.

Parameters

user_data	user data set when the signal handler was connected.

Flags: Action

The "response" signal

```
void
user_function (GtkDialog *dialog,
               gint       response_id,
               gpointer   user_data)
```

Emitted when an action widget is clicked, the dialog receives a delete event, or the application programmer calls `gtk_dialog_response()`. On a delete event, the response ID is GTK_RESPONSE_DELETE_EVENT. Otherwise, it depends on which action widget was clicked.

Parameters

dialog	the object on which the signal is emitted
response_id	the response ID
user_data	user data set when the signal handler was connected.

Flags: Run Last

See Also

GtkVBox, GtkWindow, GtkButton

GtkMessageDialog

GtkMessageDialog — A convenient message window

Functions

GtkWidget *	gtk_message_dialog_new ()
GtkWidget *	gtk_message_dialog_new_with_markup ()
void	gtk_message_dialog_set_markup ()
void	gtk_message_dialog_set_image ()
GtkWidget *	gtk_message_dialog_get_image ()
void	gtk_message_dialog_format_secondary_text ()
void	gtk_message_dialog_format_secondary_markup ()
GtkWidget *	gtk_message_dialog_get_message_area ()

Properties

GtkButtonsType	buttons	Write / Construct Only
GtkWidget *	image	Read / Write

GtkWidget *	message-area	Read
GtkMessageType	message-type	Read / Write / Construct
gchar *	secondary-text	Read / Write
gboolean	secondary-use-markup	Read / Write
gchar *	text	Read / Write
gboolean	use-markup	Read / Write

Style Properties

gint	message-border	Read

Types and Values

struct	GtkMessageDialog
enum	GtkMessageType
enum	GtkButtonsType

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkContainer
            └── GtkBin
                └── GtkWindow
                    └── GtkDialog
                        └── GtkMessageDialog
```

Implemented Interfaces

GtkMessageDialog implements AtkImplementorIface and GtkBuildable.

Includes

```
#include <gtk/gtk.h>
```

Description

GtkMessageDialog presents a dialog with some message text. It's simply a convenience widget; you could construct the equivalent of GtkMessageDialog from GtkDialog without too much effort, but GtkMessageDialog saves typing.

One difference from GtkDialog is that GtkMessageDialog sets the "skip-taskbar-hint" property to `TRUE`, so that the dialog is hidden from the taskbar by default.

The easiest way to do a modal message dialog is to use `gtk_dialog_run()`, though you can also pass in the `GTK_DIALOG_MODAL` flag, `gtk_dialog_run()` automatically makes the dialog modal and waits for the user to respond to it. `gtk_dialog_run()` returns when any dialog button is clicked.

An example for using a modal dialog:

```
1   GtkDialogFlags flags = GTK_DIALOG_DESTROY_WITH_PARENT;
```

```
  dialog = gtk_message_dialog_new (parent_window,
                                   flags,
                                   GTK_MESSAGE_ERROR,
                                   GTK_BUTTONS_CLOSE,
                                   "Error reading “%s”: %s",
                                   filename,
                                   g_strerror (errno));
  gtk_dialog_run (GTK_DIALOG (dialog));
  gtk_widget_destroy (dialog);
```

You might do a non-modal GtkMessageDialog as follows:

An example for a non-modal dialog:

```
  GtkDialogFlags flags = GTK_DIALOG_DESTROY_WITH_PARENT;
  dialog = gtk_message_dialog_new (parent_window,
                                   flags,
                                   GTK_MESSAGE_ERROR,
                                   GTK_BUTTONS_CLOSE,
                                   "Error reading “%s”: %s",
                                   filename,
                                   g_strerror (errno));

  // Destroy the dialog when the user responds to it
  // (e.g. clicks a button)

  g_signal_connect_swapped (dialog, "response",
                            G_CALLBACK (gtk_widget_destroy),
                            dialog);
```

GtkMessageDialog as GtkBuildable

The GtkMessageDialog implementation of the GtkBuildable interface exposes the message area as an internal child with the name “message_area”.

Functions

gtk_message_dialog_new ()

```
GtkWidget *
gtk_message_dialog_new (GtkWindow *parent,
                        GtkDialogFlags flags,
                        GtkMessageType type,
                        GtkButtonsType buttons,
                        const gchar *message_format,
                        ...);
```

Creates a new message dialog, which is a simple dialog with some text the user may want to see. When the user clicks a button a “response” signal is emitted with response IDs from GtkResponseType. See GtkDialog for more details.

Parameters

parent	transient parent, or `NULL` for none.	[allow-none]
flags	flags	
type	type of message	
buttons	set of buttons to use	
message_format	printf()-style format string, or `NULL`.	[allow-none]
...	arguments for `message_format`	

Returns

a new GtkMessageDialog.

[transfer none]

---

gtk_message_dialog_new_with_markup ()

```
GtkWidget *
gtk_message_dialog_new_with_markup (GtkWindow *parent,
                                    GtkDialogFlags flags,
                                    GtkMessageType type,
                                    GtkButtonsType buttons,
                                    const gchar *message_format,
                                    ...);
```

Creates a new message dialog, which is a simple dialog with some text that is marked up with the Pango text markup language. When the user clicks a button a “response” signal is emitted with response IDs from GtkResponseType. See GtkDialog for more details.

Special XML characters in the `printf()` arguments passed to this function will automatically be escaped as necessary. (See `g_markup_printf_escaped()` for how this is implemented.) Usually this is what you want, but if you have an existing Pango markup string that you want to use literally as the label, then you need to use `gtk_message_dialog_set_markup()` instead, since you can’t pass the markup string either as the format (it might contain “%” characters) or as a string argument.

```
1  GtkWidget *dialog;
2  GtkDialogFlags flags = GTK_DIALOG_DESTROY_WITH_PARENT;
3  dialog = gtk_message_dialog_new (parent_window,
4                                   flags,
5                                   GTK_MESSAGE_ERROR,
6                                   GTK_BUTTONS_CLOSE,
7                                   NULL);
8  gtk_message_dialog_set_markup (GTK_MESSAGE_DIALOG (dialog),
9                                 markup);
```

Parameters

parent	transient parent, or `NULL` for none.	[allow-none]
flags	flags	
type	type of message	
buttons	set of buttons to use	
message_format	printf()-style format string, or `NULL`.	[allow-none]
...	arguments for `message_format`	

Returns

a new GtkMessageDialog

Since: 2.4

gtk_message_dialog_set_markup ()

```
void
gtk_message_dialog_set_markup (GtkMessageDialog *message_dialog,
                               const gchar *str);
```

Sets the text of the message dialog to be `str` , which is marked up with the Pango text markup language.

Parameters

message_dialog	a GtkMessageDialog
str	markup string (see Pango markup format)

Since: 2.4

---

gtk_message_dialog_set_image ()

```
void
gtk_message_dialog_set_image (GtkMessageDialog *dialog,
                              GtkWidget *image);
```

`gtk_message_dialog_set_image` has been deprecated since version 3.12 and should not be used in newly-written code.

Use GtkDialog to create dialogs with images

Sets the dialog's image to `image` .

Parameters

dialog	a GtkMessageDialog
image	the image

Since: 2.10

---

gtk_message_dialog_get_image ()

```
GtkWidget *
gtk_message_dialog_get_image (GtkMessageDialog *dialog);
```

`gtk_message_dialog_get_image` has been deprecated since version 3.12 and should not be used in newly-written code.

Use GtkDialog for dialogs with images

Gets the dialog's image.

Parameters

dialog	a GtkMessageDialog

Returns

the dialog's image.

[transfer none]

Since: 2.14

---

gtk_message_dialog_format_secondary_text ()

```
void
gtk_message_dialog_format_secondary_text
                                (GtkMessageDialog *message_dialog,
                                 const gchar *message_format,
                                 ...);
```

Sets the secondary text of the message dialog to be `message_format` (with `printf()`-style).

Parameters

message_dialog	a GtkMessageDialog
message_format	printf()-style format string, or `NULL`. [allow-none]
...	arguments for `message_format`

Since: 2.6

---

gtk_message_dialog_format_secondary_markup ()

```
void
gtk_message_dialog_format_secondary_markup
                                (GtkMessageDialog *message_dialog,
                                 const gchar *message_format,
                                 ...);
```

Sets the secondary text of the message dialog to be `message_format` (with `printf()`-style), which is marked up with the Pango text markup language.

Due to an oversight, this function does not escape special XML characters like `gtk_message_dialog_new_with_markup()` does. Thus, if the arguments may contain special XML characters, you should use `g_markup_printf_escaped()` to escape it.

```
1  gchar *msg;
2
3  msg = g_markup_printf_escaped (message_format, ...);
4  gtk_message_dialog_format_secondary_markup (message_dialog,
5                                              "%s", msg);
6  g_free (msg);
```

Parameters

message_dialog	a GtkMessageDialog
message_format	printf()-style markup string (see Pango markup format), or `NULL`
...	arguments for `message_format`

Since: 2.6

---

gtk_message_dialog_get_message_area ()

```
GtkWidget *
gtk_message_dialog_get_message_area (GtkMessageDialog *message_dialog);
```

Returns the message area of the dialog. This is the box where the dialog's primary and secondary labels are packed. You can add your own extra content to that box and it will appear below those labels. See `gtk_dialog_get_content_area()` for the corresponding function in the parent GtkDialog.

Parameters

message_dialog	a GtkMessageDialog

Returns

A GtkBox corresponding to the "message area" in the `message_dialog` .

[transfer none]

Since: 2.22

Types and Values

struct GtkMessageDialog

```
struct GtkMessageDialog;
```

---

enum GtkMessageType

The type of message being displayed in the dialog.

Members

GTK_MESSAGE_INFO	Informational message
GTK_MESSAGE_WARNING	Non-fatal warning message
GTK_MESSAGE_QUESTION	Question requiring a choice
GTK_MESSAGE_ERROR	Fatal error message
GTK_MESSAGE_OTHER	None of the above

---

enum GtkButtonsType

Prebuilt sets of buttons for the dialog. If none of these choices are appropriate, simply use `GTK_BUTTONS_NONE` then call `gtk_dialog_add_buttons()`.

> Please note that `GTK_BUTTONS_OK`, `GTK_BUTTONS_YES_NO` and `GTK_BUTTONS_OK_CANCEL` are discouraged by the GNOME Human Interface Guidelines.

Members

GTK_BUTTONS_NONE	no buttons at all
GTK_BUTTONS_OK	an OK button
GTK_BUTTONS_CLOSE	a Close button
GTK_BUTTONS_CANCEL	a Cancel button
GTK_BUTTONS_YES_NO	Yes and No buttons
GTK_BUTTONS_OK_CANCEL	OK and Cancel buttons

Property Details

The `"buttons"` property

```
  "buttons"                  GtkButtonsType
```

The buttons shown in the message dialog.

Flags: Write / Construct Only

Default value: GTK_BUTTONS_NONE

The “image” property

“image” GtkWidget *

The image for the dialog.

GtkMessageDialog:image has been deprecated since version 3.12 and should not be used in newly-written code.

Use GtkDialog to create dialogs with images

Flags: Read / Write

Since: 2.10

The “message-area” property

“message-area” GtkWidget *

The GtkBox that corresponds to the message area of this dialog. See gtk_message_dialog_get_message_area() for a detailed description of this area.

Flags: Read

Since: 2.22

The “message-type” property

“message-type” GtkMessageType

The type of the message.

Flags: Read / Write / Construct

Default value: GTK_MESSAGE_INFO

The “secondary-text” property

“secondary-text” gchar *

The secondary text of the message dialog.

Flags: Read / Write

Default value: NULL

Since: 2.10

The “secondary-use-markup” property

“secondary-use-markup” gboolean

TRUE if the secondary text of the dialog includes Pango markup. See pango_parse_markup().

Flags: Read / Write

Default value: FALSE

Since: 2.10

The “text” property

“text” gchar *

The primary text of the message dialog. If the dialog has a secondary text, this will appear as the title.

Flags: Read / Write

Default value: ""

Since: 2.10

The “use-markup” property

“use-markup” gboolean

TRUE if the primary text of the dialog includes Pango markup. See pango_parse_markup().

Flags: Read / Write

Default value: FALSE

Since: 2.10

Style Property Details

The “message-border” style property

“message-border” gint

Width of border around the label in the message dialog.

Flags: Read

Allowed values: >= 0

Default value: 12

See Also

GtkDialog

GtkAboutDialog

GtkAboutDialog — Display information about an application

Functions

GtkWidget *	gtk_about_dialog_new ()
const gchar *	gtk_about_dialog_get_program_name ()
void	gtk_about_dialog_set_program_name ()
const gchar *	gtk_about_dialog_get_version ()
void	gtk_about_dialog_set_version ()
const gchar *	gtk_about_dialog_get_copyright ()
void	gtk_about_dialog_set_copyright ()
const gchar *	gtk_about_dialog_get_comments ()
void	gtk_about_dialog_set_comments ()
const gchar *	gtk_about_dialog_get_license ()
void	gtk_about_dialog_set_license ()
gboolean	gtk_about_dialog_get_wrap_license ()
void	gtk_about_dialog_set_wrap_license ()
GtkLicense	gtk_about_dialog_get_license_type ()
void	gtk_about_dialog_set_license_type ()
const gchar *	gtk_about_dialog_get_website ()
void	gtk_about_dialog_set_website ()
const gchar *	gtk_about_dialog_get_website_label ()
void	gtk_about_dialog_set_website_label ()
const gchar * const *	gtk_about_dialog_get_authors ()
void	gtk_about_dialog_set_authors ()
const gchar * const *	gtk_about_dialog_get_artists ()
void	gtk_about_dialog_set_artists ()
const gchar * const *	gtk_about_dialog_get_documenters ()
void	gtk_about_dialog_set_documenters ()
const gchar *	gtk_about_dialog_get_translator_credits ()
void	gtk_about_dialog_set_translator_credits ()
GdkPixbuf *	gtk_about_dialog_get_logo ()
void	gtk_about_dialog_set_logo ()
const gchar *	gtk_about_dialog_get_logo_icon_name ()
void	gtk_about_dialog_set_logo_icon_name ()

void	gtk_about_dialog_add_credit_section ()
void	gtk_show_about_dialog ()

Properties

GStrv	artists	Read / Write
GStrv	authors	Read / Write
gchar *	comments	Read / Write
gchar *	copyright	Read / Write
GStrv	documenters	Read / Write
gchar *	license	Read / Write
GtkLicense	license-type	Read / Write
GdkPixbuf *	logo	Read / Write
gchar *	logo-icon-name	Read / Write
gchar *	program-name	Read / Write
gchar *	translator-credits	Read / Write
gchar *	version	Read / Write
gchar *	website	Read / Write
gchar *	website-label	Read / Write
gboolean	wrap-license	Read / Write

Signals

gboolean	activate-link	Run Last

Types and Values

struct	GtkAboutDialog
enum	GtkLicense

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkContainer
            └── GtkBin
                └── GtkWindow
                    └── GtkDialog
                        └── GtkAboutDialog
```

Implemented Interfaces

GtkAboutDialog implements AtkImplementorIface and GtkBuildable.

Includes

```
#include <gtk/gtk.h>
```

Description

The GtkAboutDialog offers a simple way to display information about a program like its logo, name, copyright, website and license. It is also possible to give credits to the authors, documenters, translators and artists who have worked on the program. An about dialog is typically opened when the user selects the `About` option from the `Help` menu. All parts of the dialog are optional.

About dialogs often contain links and email addresses. GtkAboutDialog displays these as clickable links. By default, it calls `gtk_show_uri_on_window()` when a user clicks one. The behaviour can be overridden with the "activate-link" signal.

To specify a person with an email address, use a string like "Edgar Allan Poe <edgar@poe.com>". To specify a website with a title, use a string like "GTK+ team http://www.gtk.org".

To make constructing a GtkAboutDialog as convenient as possible, you can use the function `gtk_show_about_dialog()` which constructs and shows a dialog and keeps it around so that it can be shown again.

Note that GTK+ sets a default title of `_("About %s")` on the dialog window (where %s is replaced by the name of the application, but in order to ensure proper translation of the title, applications should set the title property explicitly when constructing a GtkAboutDialog, as shown in the following example:

```
1   GdkPixbuf *example_logo = gdk_pixbuf_new_from_file ("./logo.png", NULL);
2   gtk_show_about_dialog (NULL,
3                          "program-name", "ExampleCode",
4                          "logo", example_logo,
5                          "title", _("About ExampleCode"),
6                          NULL);
```

It is also possible to show a GtkAboutDialog like any other GtkDialog, e.g. using `gtk_dialog_run()`. In this case, you might need to know that the "Close" button returns the GTK_RESPONSE_CANCEL response id.

Functions

gtk_about_dialog_new ()

```
GtkWidget *
gtk_about_dialog_new (void);
```

Creates a new GtkAboutDialog.

Returns

a newly created GtkAboutDialog

Since: 2.6

---

gtk_about_dialog_get_program_name ()

```
const gchar *
gtk_about_dialog_get_program_name (GtkAboutDialog *about);
```

Returns the program name displayed in the about dialog.

Parameters

about	a GtkAboutDialog

Returns

The program name. The string is owned by the about dialog and must not be modified.

Since: 2.12

---

gtk_about_dialog_set_program_name ()

```
void
gtk_about_dialog_set_program_name (GtkAboutDialog *about,
                                   const gchar *name);
```

Sets the name to display in the about dialog. If this is not set, it defaults to `g_get_application_name()`.

Parameters

about	a GtkAboutDialog
name	the program name

Since: 2.12

---

gtk_about_dialog_get_version ()

```
const gchar *
gtk_about_dialog_get_version (GtkAboutDialog *about);
```

Returns the version string.

Parameters

about	a GtkAboutDialog

Returns

The version string. The string is owned by the about dialog and must not be modified.

Since: 2.6

---

gtk_about_dialog_set_version ()

```
void
gtk_about_dialog_set_version (GtkAboutDialog *about,
                              const gchar *version);
```

Sets the version string to display in the about dialog.

Parameters

about	a GtkAboutDialog	
version	the version string.	[allow-none]

Since: 2.6

---

gtk_about_dialog_get_copyright ()

```
const gchar *
gtk_about_dialog_get_copyright (GtkAboutDialog *about);
```

Returns the copyright string.

Parameters

about	a GtkAboutDialog	

Returns

The copyright string. The string is owned by the about dialog and must not be modified.

Since: 2.6

---

gtk_about_dialog_set_copyright ()

```
void
gtk_about_dialog_set_copyright (GtkAboutDialog *about,
                                const gchar *copyright);
```

Sets the copyright string to display in the about dialog. This should be a short string of one or two lines.

Parameters

about	a GtkAboutDialog	
copyright	the copyright string.	[allow-none]

Since: 2.6

---

gtk_about_dialog_get_comments ()

```
const gchar *
gtk_about_dialog_get_comments (GtkAboutDialog *about);
```

Returns the comments string.

Parameters

about	a GtkAboutDialog	

Returns

The comments. The string is owned by the about dialog and must not be modified.

Since: 2.6

---

gtk_about_dialog_set_comments ()

```
void
gtk_about_dialog_set_comments (GtkAboutDialog *about,
                               const gchar *comments);
```

Sets the comments string to display in the about dialog. This should be a short string of one or two lines.

Parameters

about	a GtkAboutDialog	
comments	a comments string.	[allow-none]

Since: 2.6

---

gtk_about_dialog_get_license ()

```
const gchar *
gtk_about_dialog_get_license (GtkAboutDialog *about);
```

Returns the license information.

Parameters

about	a GtkAboutDialog	

Returns

The license information. The string is owned by the about dialog and must not be modified.

Since: 2.6

---

gtk_about_dialog_set_license ()

```
void
gtk_about_dialog_set_license (GtkAboutDialog *about,
                              const gchar *license);
```

Sets the license information to be displayed in the secondary license dialog. If `license` is `NULL`, the license button is hidden.

Parameters

about	a GtkAboutDialog	
license	the license information or `NULL`.	[allow-none]

Since: 2.6

---

gtk_about_dialog_get_wrap_license ()

```
gboolean
gtk_about_dialog_get_wrap_license (GtkAboutDialog *about);
```

Returns whether the license text in `about` is automatically wrapped.

Parameters

about	a GtkAboutDialog

Returns

`TRUE` if the license text is wrapped

Since: 2.8

---

gtk_about_dialog_set_wrap_license ()

```
void
gtk_about_dialog_set_wrap_license (GtkAboutDialog *about,
                                   gboolean wrap_license);
```

Sets whether the license text in `about` is automatically wrapped.

Parameters

about	a GtkAboutDialog
wrap_license	whether to wrap the license

Since: 2.8

---

gtk_about_dialog_get_license_type ()

```
GtkLicense
gtk_about_dialog_get_license_type (GtkAboutDialog *about);
```

Retrieves the license set using `gtk_about_dialog_set_license_type()`

Parameters

about	a GtkAboutDialog

Returns

a GtkLicense value

Since: 3.0

---

gtk_about_dialog_set_license_type ()

```
void
gtk_about_dialog_set_license_type (GtkAboutDialog *about,
                                   GtkLicense license_type);
```

Sets the license of the application showing the `about` dialog from a list of known licenses.

This function overrides the license set using `gtk_about_dialog_set_license()`.

Parameters

about	a GtkAboutDialog
license_type	the type of license

Since: 3.0

---

gtk_about_dialog_get_website ()

```
const gchar *
gtk_about_dialog_get_website (GtkAboutDialog *about);
```

Returns the website URL.

Parameters

about	a GtkAboutDialog

Returns

The website URL. The string is owned by the about dialog and must not be modified.

Since: 2.6

---

gtk_about_dialog_set_website ()

```
void
gtk_about_dialog_set_website (GtkAboutDialog *about,
                              const gchar *website);
```

Sets the URL to use for the website link.

Parameters

about	a GtkAboutDialog
website	a URL string starting with "http://". [allow-none]

Since: 2.6

---

gtk_about_dialog_get_website_label ()

```
const gchar *
gtk_about_dialog_get_website_label (GtkAboutDialog *about);
```

Returns the label used for the website link.

Parameters

about	a GtkAboutDialog

Returns

The label used for the website link. The string is owned by the about dialog and must not be modified.

Since: 2.6

---

gtk_about_dialog_set_website_label ()

```
void
gtk_about_dialog_set_website_label (GtkAboutDialog *about,
                                    const gchar *website_label);
```

Sets the label to be used for the website link.

Parameters

about	a GtkAboutDialog
website_label	the label used for the website link

Since: 2.6

---

gtk_about_dialog_get_authors ()

```
const gchar * const *
gtk_about_dialog_get_authors (GtkAboutDialog *about);
```

Returns the string which are displayed in the authors tab of the secondary credits dialog.

Parameters

about	a GtkAboutDialog

Returns

A `NULL`-terminated string array containing the authors. The array is owned by the about dialog and must not be modified.

[array zero-terminated=1][transfer none]

Since: 2.6

---

gtk_about_dialog_set_authors ()

```
void
gtk_about_dialog_set_authors (GtkAboutDialog *about,
                              const gchar **authors);
```

Sets the strings which are displayed in the authors tab of the secondary credits dialog.

Parameters

about	a GtkAboutDialog	
authors	a `NULL`-terminated array of strings.	[array zero-terminated=1]

Since: 2.6

---

gtk_about_dialog_get_artists ()

```
const gchar * const *
gtk_about_dialog_get_artists (GtkAboutDialog *about);
```

Returns the string which are displayed in the artists tab of the secondary credits dialog.

Parameters

about	a GtkAboutDialog

Returns

A `NULL`-terminated string array containing the artists. The array is owned by the about dialog and must not be modified.

[array zero-terminated=1][transfer none]

Since: 2.6

---

gtk_about_dialog_set_artists ()

```
void
gtk_about_dialog_set_artists (GtkAboutDialog *about,
                              const gchar **artists);
```

Sets the strings which are displayed in the artists tab of the secondary credits dialog.

Parameters

about	a GtkAboutDialog	
artists	a `NULL`-terminated array of strings.	[array zero-terminated=1]

Since: 2.6

---

gtk_about_dialog_get_documenters ()

```
const gchar * const *
gtk_about_dialog_get_documenters (GtkAboutDialog *about);
```

Returns the string which are displayed in the documenters tab of the secondary credits dialog.

Parameters

about	a GtkAboutDialog

Returns

A `NULL`-terminated string array containing the documenters. The array is owned by the about dialog and must not be modified.

[array zero-terminated=1][transfer none]

Since: 2.6

---

gtk_about_dialog_set_documenters ()

```
void
gtk_about_dialog_set_documenters (GtkAboutDialog *about,
                                  const gchar **documenters);
```

Sets the strings which are displayed in the documenters tab of the secondary credits dialog.

Parameters

about	a GtkAboutDialog	
documenters	a `NULL`-terminated array of strings.	[array zero-terminated=1]

Since: 2.6

---

gtk_about_dialog_get_translator_credits ()

```
const gchar *
gtk_about_dialog_get_translator_credits
                                (GtkAboutDialog *about);
```

Returns the translator credits string which is displayed in the translators tab of the secondary credits dialog.

Parameters

about	a GtkAboutDialog	

Returns

The translator credits string. The string is owned by the about dialog and must not be modified.

Since: 2.6

---

gtk_about_dialog_set_translator_credits ()

```
void
gtk_about_dialog_set_translator_credits
                                (GtkAboutDialog *about,
                                 const gchar *translator_credits);
```

Sets the translator credits string which is displayed in the translators tab of the secondary credits dialog.

The intended use for this string is to display the translator of the language which is currently used in the user interface. Using `gettext()`, a simple way to achieve that is to mark the string for translation:

```
1    GtkWidget *about = gtk_about_dialog_new ();
2    gtk_about_dialog_set_translator_credits (GTK_ABOUT_DIALOG (about),
3                                             _("translator-credits"));
```

It is a good idea to use the customary msgid “translator-credits” for this purpose, since translators will already know the purpose of that msgid, and since GtkAboutDialog will detect if “translator-credits” is untranslated and hide the tab.

Parameters

about	a GtkAboutDialog	
translator_credits	the translator credits.	[allow-none]

Since: 2.6

---

gtk_about_dialog_get_logo ()

```
GdkPixbuf *
gtk_about_dialog_get_logo (GtkAboutDialog *about);
```

Returns the pixbuf displayed as logo in the about dialog.

Parameters

about	a GtkAboutDialog	

Returns

the pixbuf displayed as logo. The pixbuf is owned by the about dialog. If you want to keep a reference to it, you have to call `g_object_ref()` on it.

[transfer none]

Since: 2.6

---

gtk_about_dialog_set_logo ()

```
void
gtk_about_dialog_set_logo (GtkAboutDialog *about,
                           GdkPixbuf *logo);
```

Sets the pixbuf to be displayed as logo in the about dialog. If it is `NULL`, the default window icon set with `gtk_window_set_default_icon()` will be used.

Parameters

about	a GtkAboutDialog	
logo	a GdkPixbuf, or `NULL`.	[allow-none]

Since: 2.6

---

gtk_about_dialog_get_logo_icon_name ()

```
const gchar *
gtk_about_dialog_get_logo_icon_name (GtkAboutDialog *about);
```

Returns the icon name displayed as logo in the about dialog.

Parameters

about	a GtkAboutDialog	

Returns

the icon name displayed as logo. The string is owned by the dialog. If you want to keep a reference to it, you have to call `g_strdup()` on it.

Since: 2.6

---

gtk_about_dialog_set_logo_icon_name ()

```
void
gtk_about_dialog_set_logo_icon_name (GtkAboutDialog *about,
                                     const gchar *icon_name);
```

Sets the pixbuf to be displayed as logo in the about dialog. If it is NULL, the default window icon set with gtk_window_set_default_icon() will be used.

Parameters

about	a GtkAboutDialog	
icon_name	an icon name, or NULL.	[allow-none]

Since: 2.6

---

gtk_about_dialog_add_credit_section ()

```
void
gtk_about_dialog_add_credit_section (GtkAboutDialog *about,
                                     const gchar *section_name,
                                     const gchar **people);
```

Creates a new section in the Credits page.

Parameters

about	A GtkAboutDialog	
section_name	The name of the section	
people	The people who belong to that section.	[array zero-terminated=1]

Since: 3.4

---

gtk_show_about_dialog ()

```
void
gtk_show_about_dialog (GtkWindow *parent,
                       const gchar *first_property_name,
                       ...);
```

This is a convenience function for showing an application's about box. The constructed dialog is associated with the parent window and reused for future invocations of this function.

Parameters

parent	transient parent, or NULL for none.	[allow-none]
first_property_name	the name of the first property	
...	value of first property, followed by more properties, NULL-terminated	

Since: 2.6

Types and Values

struct GtkAboutDialog

```
struct GtkAboutDialog;
```

The GtkAboutDialog contains only private fields and should not be directly accessed.

---

enum GtkLicense

The type of license for an application.

This enumeration can be expanded at later date.

Members

GTK_LICENSE_UNKNOWN	No license specified
GTK_LICENSE_CUSTOM	A license text is going to be specified by the developer
GTK_LICENSE_GPL_2_0	The GNU General Public License, version 2.0 or later
GTK_LICENSE_GPL_3_0	The GNU General Public License, version 3.0 or later
GTK_LICENSE_LGPL_2_1	The GNU Lesser General Public License, version 2.1 or later
GTK_LICENSE_LGPL_3_0	The GNU Lesser General Public License, version 3.0 or later
GTK_LICENSE_BSD	The BSD standard license
GTK_LICENSE_MIT_X11	The MIT/X11 standard license
GTK_LICENSE_ARTISTIC	The Artistic License, version 2.0
GTK_LICENSE_GPL_2_0_ONLY	The GNU General Public License, version 2.0 only. Since 3.12.
GTK_LICENSE_GPL_3_0_ONLY	The GNU General Public License, version 3.0 only. Since 3.12.
GTK_LICENSE_LGPL_2_1_ONLY	The GNU Lesser General Public License, version 2.1 only. Since 3.12.
GTK_LICENSE_LGPL_3_0_ONLY	The GNU Lesser General Public License, version 3.0 only. Since 3.12.
GTK_LICENSE_AGPL_3_0	The GNU Affero General Public License, version 3.0 or later. Since: 3.22.
GTK_LICENSE_AGPL_3_0_ONLY	The GNU Affero General Public License, version 3.0 only. Since: 3.22.27.

Since: 3.0

Property Details

The “artists” property

```
  “artists”                  GStrv
```

The people who contributed artwork to the program, as a NULL-terminated array of strings. Each string may contain email addresses and URLs, which will be displayed as links, see the introduction for more details.

Flags: Read / Write

Since: 2.6

The “authors” property

“authors” GStrv

The authors of the program, as a NULL-terminated array of strings. Each string may contain email addresses and URLs, which will be displayed as links, see the introduction for more details.

Flags: Read / Write

Since: 2.6

The “comments” property

“comments” gchar *

Comments about the program. This string is displayed in a label in the main dialog, thus it should be a short explanation of the main purpose of the program, not a detailed list of features.

Flags: Read / Write

Default value: NULL

Since: 2.6

The “copyright” property

“copyright” gchar *

Copyright information for the program.

Flags: Read / Write

Default value: NULL

Since: 2.6

The “documenters” property

“documenters” GStrv

The people documenting the program, as a NULL-terminated array of strings. Each string may contain email addresses and URLs, which will be displayed as links, see the introduction for more details.

Flags: Read / Write

Since: 2.6

The “license” property

“license” gchar *

The license of the program. This string is displayed in a text view in a secondary dialog, therefore it is fine to use a long multi-paragraph text. Note that the text is only wrapped in the text view if the "wrap-license" property is set to `TRUE`; otherwise the text itself must contain the intended linebreaks. When setting this property to a non-`NULL` value, the "license-type" property is set to `GTK_LICENSE_CUSTOM` as a side effect.

Flags: Read / Write

Default value: NULL

Since: 2.6

The "license-type" property

```
"license-type"              GtkLicense
```

The license of the program, as a value of the `GtkLicense` enumeration.

The GtkAboutDialog will automatically fill out a standard disclaimer and link the user to the appropriate online resource for the license text.

If `GTK_LICENSE_UNKNOWN` is used, the link used will be the same specified in the "website" property.

If `GTK_LICENSE_CUSTOM` is used, the current contents of the "license" property are used.

For any other GtkLicense value, the contents of the "license" property are also set by this property as a side effect.

Flags: Read / Write

Default value: GTK_LICENSE_UNKNOWN

Since: 3.0

The "logo" property

```
"logo"                      GdkPixbuf *
```

A logo for the about box. If it is `NULL`, the default window icon set with `gtk_window_set_default_icon()` will be used.

Flags: Read / Write

Since: 2.6

The "logo-icon-name" property

```
"logo-icon-name"            gchar *
```

A named icon to use as the logo for the about box. This property overrides the "logo" property.

Flags: Read / Write

Default value: "image-missing"

Since: 2.6

The “program-name” property

“program-name” gchar *

The name of the program. If this is not set, it defaults to g_get_application_name().

Flags: Read / Write

Default value: NULL

Since: 2.12

The “translator-credits” property

“translator-credits” gchar *

Credits to the translators. This string should be marked as translatable. The string may contain email addresses and URLs, which will be displayed as links, see the introduction for more details.

Flags: Read / Write

Default value: NULL

Since: 2.6

The “version” property

“version” gchar *

The version of the program.

Flags: Read / Write

Default value: NULL

Since: 2.6

The “website” property

“website” gchar *

The URL for the link to the website of the program. This should be a string starting with "http://.

Flags: Read / Write

Default value: NULL

Since: 2.6

The “website-label” property

```
  “website-label”            gchar *
```

The label for the link to the website of the program.

Flags: Read / Write

Default value: NULL

Since: 2.6

The “wrap-license” property

```
  “wrap-license”             gboolean
```

Whether to wrap the text in the license dialog.

Flags: Read / Write

Default value: FALSE

Since: 2.8

Signal Details

The “activate-link” signal

```
gboolean
user_function (GtkAboutDialog *label,
               gchar          *uri,
               gpointer        user_data)
```

The signal which gets emitted to activate a URI. Applications may connect to it to override the default behaviour, which is to call `gtk_show_uri_on_window()`.

Parameters

label	The object on which the signal was emitted
uri	the URI that is activated
user_data	user data set when the signal handler was connected.

Returns

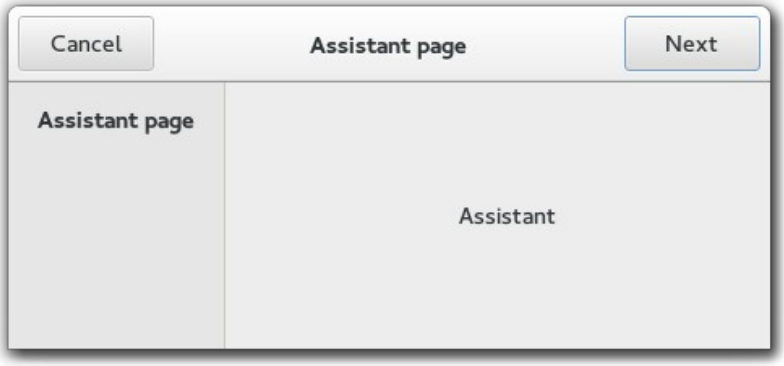
`TRUE` if the link has been activated

Flags: Run Last

Since: 2.24

GtkAssistant

GtkAssistant — A widget used to guide users through multi-step operations



Functions

GtkWidget *	gtk_assistant_new ()
gint	gtk_assistant_get_current_page ()
void	gtk_assistant_set_current_page ()
gint	gtk_assistant_get_n_pages ()
GtkWidget *	gtk_assistant_get_nth_page ()
gint	gtk_assistant_prepend_page ()
gint	gtk_assistant_append_page ()
gint	gtk_assistant_insert_page ()
void	gtk_assistant_remove_page ()
gint	(*GtkAssistantPageFunc) ()
void	gtk_assistant_set_forward_page_func ()
void	gtk_assistant_set_page_type ()
GtkAssistantPageType	gtk_assistant_get_page_type ()
void	gtk_assistant_set_page_title ()
const gchar *	gtk_assistant_get_page_title ()
void	gtk_assistant_set_page_header_image ()
GdkPixbuf *	gtk_assistant_get_page_header_image ()
void	gtk_assistant_set_page_side_image ()
GdkPixbuf *	gtk_assistant_get_page_side_image ()
void	gtk_assistant_set_page_complete ()
gboolean	gtk_assistant_get_page_complete ()
void	gtk_assistant_set_page_has_padding ()
gboolean	gtk_assistant_get_page_has_padding ()
void	gtk_assistant_add_action_widget ()
void	gtk_assistant_remove_action_widget ()
void	gtk_assistant_update_buttons_state ()
void	gtk_assistant_commit ()
void	gtk_assistant_next_page ()
void	gtk_assistant_previous_page ()

Properties

gint	use-header-bar	Read / Write / Construct Only

Child Properties

gboolean	complete	Read / Write
gboolean	has-padding	Read / Write
GdkPixbuf *	header-image	Read / Write
GtkAssistantPageType	page-type	Read / Write
GdkPixbuf *	sidebar-image	Read / Write
gchar *	title	Read / Write

Style Properties

gint	content-padding	Read
gint	header-padding	Read

Signals

void	apply	Run Last
void	cancel	Run Last
void	close	Run Last
void	escape	Action
void	prepare	Run Last

Types and Values

struct	GtkAssistant
struct	GtkAssistantClass
enum	GtkAssistantPageType

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkContainer
            └── GtkBin
                └── GtkWindow
                    └── GtkAssistant
```

Implemented Interfaces

GtkAssistant implements AtkImplementorIface and GtkBuildable.

Includes

```
#include <gtk/gtk.h>
```

Description

A GtkAssistant is a widget used to represent a generally complex operation splitted in several steps, guiding the user through its pages and controlling the page flow to collect the necessary data.

The design of GtkAssistant is that it controls what buttons to show and to make sensitive, based on what it knows about the page sequence and the type of each page, in addition to state information like the page completion and committed status.

If you have a case that doesn't quite fit in GtkAssistants way of handling buttons, you can use the GTK_ASSISTANT_PAGE_CUSTOM page type and handle buttons yourself.

GtkAssistant as GtkBuildable

The GtkAssistant implementation of the GtkBuildable interface exposes the `action_area` as internal children with the name "action_area".

To add pages to an assistant in GtkBuilder, simply add it as a child to the GtkAssistant object, and set its child properties as necessary.

CSS nodes

GtkAssistant has a single CSS node with the name assistant.

Functions

gtk_assistant_new ()

```
GtkWidget *
gtk_assistant_new (void);
```

Creates a new GtkAssistant.

Returns

a newly created GtkAssistant

Since: 2.10

---

gtk_assistant_get_current_page ()

```
gint
gtk_assistant_get_current_page (GtkAssistant *assistant);
```

Returns the page number of the current page.

Parameters

assistant	a GtkAssistant

Returns

The index (starting from 0) of the current page in the `assistant` , or -1 if the `assistant` has no pages, or no current page.

Since: 2.10

---

gtk_assistant_set_current_page ()

```
void
gtk_assistant_set_current_page (GtkAssistant *assistant,
                                gint page_num);
```

Switches the page to `page_num` .

Note that this will only be necessary in custom buttons, as the `assistant` flow can be set with `gtk_assistant_set_forward_page_func()`.

Parameters

assistant	a GtkAssistant
page_num	index of the page to switch to, starting from 0. If negative, the last page will be used. If greater than the number of pages in the `assistant` , nothing will be done.

Since: 2.10

---

gtk_assistant_get_n_pages ()

```
gint
gtk_assistant_get_n_pages (GtkAssistant *assistant);
```

Returns the number of pages in the `assistant`

Parameters

assistant	a GtkAssistant

Returns

the number of pages in the `assistant`

Since: 2.10

---

gtk_assistant_get_nth_page ()

```
GtkWidget *
gtk_assistant_get_nth_page (GtkAssistant *assistant,
                            gint page_num);
```

Returns the child widget contained in page number `page_num` .

Parameters

assistant	a GtkAssistant
page_num	the index of a page in the `assistant` , or -1 to get the last page

Returns

the child widget, or `NULL` if `page_num` is out of bounds.

[nullable][transfer none]

Since: 2.10

---

gtk_assistant_prepend_page ()

```
gint
gtk_assistant_prepend_page (GtkAssistant *assistant,
                            GtkWidget *page);
```

Prepends a page to the `assistant` .

Parameters

assistant	a GtkAssistant
page	a GtkWidget

Returns

the index (starting at 0) of the inserted page

Since: 2.10

---

gtk_assistant_append_page ()

```
gint
gtk_assistant_append_page (GtkAssistant *assistant,
                           GtkWidget *page);
```

Appends a page to the `assistant` .

Parameters

assistant	a GtkAssistant
page	a GtkWidget

Returns

the index (starting at 0) of the inserted page

Since: 2.10

---

gtk_assistant_insert_page ()

```
gint
gtk_assistant_insert_page (GtkAssistant *assistant,
                           GtkWidget *page,
                           gint position);
```

Inserts a page in the `assistant` at a given position.

Parameters

assistant	a GtkAssistant
page	a GtkWidget
position	the index (starting at 0) at which to insert the page, or -1 to append the page to the `assistant`

Returns

the index (starting from 0) of the inserted page

Since: 2.10

---

gtk_assistant_remove_page ()

```
void
gtk_assistant_remove_page (GtkAssistant *assistant,
                           gint page_num);
```

Removes the `page_num` 's page from `assistant` .

Parameters

assistant	a GtkAssistant
page_num	the index of a page in the `assistant` , or -1 to remove the last page

Since: 3.2

GtkAssistantPageFunc ()

```
gint
(*GtkAssistantPageFunc) (gint current_page,
                         gpointer data);
```

A function used by `gtk_assistant_set_forward_page_func()` to know which is the next page given a current one. It's called both for computing the next page when the user presses the "forward" button and for handling the behavior of the "last" button.

Parameters

current_page	The page number used to calculate the next page.	
data	user data.	[closure]

Returns

The next page number.

---

gtk_assistant_set_forward_page_func ()

```
void
gtk_assistant_set_forward_page_func (GtkAssistant *assistant,
                                     GtkAssistantPageFunc page_func,
                                     gpointer data,
                                     GDestroyNotify destroy);
```

Sets the page forwarding function to be `page_func` .

This function will be used to determine what will be the next page when the user presses the forward button. Setting `page_func` to `NULL` will make the assistant to use the default forward function, which just goes to the next visible page.

Parameters

assistant	a GtkAssistant	
page_func	the GtkAssistantPageFunc, or `NULL` to use the default one.	[allow-none]
data	user data for `page_func`	
destroy	destroy notifier for `data`	

Since: 2.10

---

gtk_assistant_set_page_type ()

```
void
gtk_assistant_set_page_type (GtkAssistant *assistant,
                             GtkWidget *page,
                             GtkAssistantPageType type);
```

Sets the page type for `page` .

The page type determines the page behavior in the `assistant` .

Parameters

assistant	a GtkAssistant
page	a page of `assistant`
type	the new type for `page`

Since: 2.10

---

gtk_assistant_get_page_type ()

```
GtkAssistantPageType
gtk_assistant_get_page_type (GtkAssistant *assistant,
                             GtkWidget *page);
```

Gets the page type of `page` .

Parameters

assistant	a GtkAssistant
page	a page of `assistant`

Returns

the page type of `page`

Since: 2.10

---

gtk_assistant_set_page_title ()

```
void
gtk_assistant_set_page_title (GtkAssistant *assistant,
                              GtkWidget *page,
                              const gchar *title);
```

Sets a title for `page` .

The title is displayed in the header area of the assistant when `page` is the current page.

Parameters

assistant	a GtkAssistant
page	a page of `assistant`
title	the new title for `page`

Since: 2.10

---

gtk_assistant_get_page_title ()

```
const gchar *
gtk_assistant_get_page_title (GtkAssistant *assistant,
                              GtkWidget *page);
```

Gets the title for `page` .

Parameters

assistant	a GtkAssistant
page	a page of `assistant`

Returns

the title for `page`

Since: 2.10

---

gtk_assistant_set_page_header_image ()

```
void
gtk_assistant_set_page_header_image (GtkAssistant *assistant,
                                     GtkWidget *page,
                                     GdkPixbuf *pixbuf);
```

`gtk_assistant_set_page_header_image` has been deprecated since version 3.2 and should not be used in newly-written code.

Since GTK+ 3.2, a header is no longer shown; add your header decoration to the page content instead.

Sets a header image for `page` .

Parameters

assistant	a GtkAssistant	
page	a page of `assistant`	
pixbuf	the new header image `page` .	[allow-none]

Since: 2.10

---

gtk_assistant_get_page_header_image ()

```
GdkPixbuf *
gtk_assistant_get_page_header_image (GtkAssistant *assistant,
                                     GtkWidget *page);
```

`gtk_assistant_get_page_header_image` has been deprecated since version 3.2 and should not be used in newly-written code.

Since GTK+ 3.2, a header is no longer shown; add your header decoration to the page content instead.

Gets the header image for `page` .

Parameters

assistant	a GtkAssistant
page	a page of `assistant`

Returns

the header image for `page` , or `NULL` if there’s no header image for the page.

[transfer none]

Since: 2.10

---

gtk_assistant_set_page_side_image ()

```
void
gtk_assistant_set_page_side_image (GtkAssistant *assistant,
                                   GtkWidget *page,
                                   GdkPixbuf *pixbuf);
```

`gtk_assistant_set_page_side_image` has been deprecated since version 3.2 and should not be used in newly-written code.

Since GTK+ 3.2, sidebar images are not shown anymore.

Sets a side image for `page` .

This image used to be displayed in the side area of the assistant when `page` is the current page.

Parameters

assistant	a GtkAssistant	
page	a page of `assistant`	
pixbuf	the new side image `page` .	[allow-none]

Since: 2.10

---

gtk_assistant_get_page_side_image ()

```
GdkPixbuf *
gtk_assistant_get_page_side_image (GtkAssistant *assistant,
                                   GtkWidget *page);
```

`gtk_assistant_get_page_side_image` has been deprecated since version 3.2 and should not be used in newly-written code.

Since GTK+ 3.2, sidebar images are not shown anymore.

Gets the side image for `page` .

Parameters

assistant	a GtkAssistant
page	a page of `assistant`

Returns

the side image for `page` , or `NULL` if there’s no side image for the page.

[transfer none]

Since: 2.10

---

gtk_assistant_set_page_complete ()

```
void
gtk_assistant_set_page_complete (GtkAssistant *assistant,
                                 GtkWidget *page,
                                 gboolean complete);
```

Sets whether `page` contents are complete.

This will make `assistant` update the buttons state to be able to continue the task.

Parameters

assistant	a GtkAssistant
page	a page of `assistant`
complete	the completeness status of the page

Since: 2.10

---

gtk_assistant_get_page_complete ()

```
gboolean
gtk_assistant_get_page_complete (GtkAssistant *assistant,
                                 GtkWidget *page);
```

Gets whether `page` is complete.

Parameters

assistant	a GtkAssistant
page	a page of `assistant`

Returns

`TRUE` if `page` is complete.

Since: 2.10

---

gtk_assistant_set_page_has_padding ()

```
void
gtk_assistant_set_page_has_padding (GtkAssistant *assistant,
                                    GtkWidget *page,
                                    gboolean has_padding);
```

Sets whether the assistant is adding padding around the page.

Parameters

assistant	a GtkAssistant
page	a page of `assistant`
has_padding	whether this page has padding

Since: 3.18

---

gtk_assistant_get_page_has_padding ()

```
gboolean
gtk_assistant_get_page_has_padding (GtkAssistant *assistant,
                                    GtkWidget *page);
```

Gets whether page has padding.

Parameters

assistant	a GtkAssistant
page	a page of `assistant`

Returns

`TRUE` if `page` has padding

Since: 3.18

---

gtk_assistant_add_action_widget ()

```
void
gtk_assistant_add_action_widget (GtkAssistant *assistant,
                                 GtkWidget *child);
```

Adds a widget to the action area of a GtkAssistant.

Parameters

assistant	a GtkAssistant
child	a GtkWidget

Since: 2.10

---

gtk_assistant_remove_action_widget ()

```
void
gtk_assistant_remove_action_widget (GtkAssistant *assistant,
                                    GtkWidget *child);
```

Removes a widget from the action area of a GtkAssistant.

Parameters

assistant	a GtkAssistant
child	a GtkWidget

Since: 2.10

---

gtk_assistant_update_buttons_state ()

```
void
gtk_assistant_update_buttons_state (GtkAssistant *assistant);
```

Forces `assistant` to recompute the buttons state.

GTK+ automatically takes care of this in most situations, e.g. when the user goes to a different page, or when the visibility or completeness of a page changes.

One situation where it can be necessary to call this function is when changing a value on the current page affects the future page flow of the assistant.

Parameters

assistant	a GtkAssistant

Since: 2.10

---

gtk_assistant_commit ()

```
void
gtk_assistant_commit (GtkAssistant *assistant);
```

Erases the visited page history so the back button is not shown on the current page, and removes the cancel button from subsequent pages.

Use this when the information provided up to the current page is hereafter deemed permanent and cannot be modified or undone. For example, showing a progress page to track a long-running, unreversible operation after the user has clicked apply on a confirmation page.

Parameters

assistant	a GtkAssistant

Since: 2.22

---

gtk_assistant_next_page ()

```
void
gtk_assistant_next_page (GtkAssistant *assistant);
```

Navigate to the next page.

It is a programming error to call this function when there is no next page.

This function is for use when creating pages of the GTK_ASSISTANT_PAGE_CUSTOM type.

Parameters

assistant	a GtkAssistant

Since: 3.0

---

gtk_assistant_previous_page ()

```
void
gtk_assistant_previous_page (GtkAssistant *assistant);
```

Navigate to the previous visited page.

It is a programming error to call this function when no previous page is available.

This function is for use when creating pages of the GTK_ASSISTANT_PAGE_CUSTOM type.

Parameters

assistant	a GtkAssistant

Since: 3.0

Types and Values

struct GtkAssistant

```
struct GtkAssistant;
```

---

struct GtkAssistantClass

```
struct GtkAssistantClass {
  GtkWindowClass parent_class;


  void (* prepare) (GtkAssistant *assistant, GtkWidget *page);
  void (* apply)   (GtkAssistant *assistant);
  void (* close)   (GtkAssistant *assistant);
  void (* cancel)  (GtkAssistant *assistant);
};
```

Members

`prepare` ()	Signal emitted when a new page is set as the assistant's current page, before making the new page visible.
`apply` ()	Signal emitted when the apply button is clicked.
`close` ()	Signal emitted either when the close button or last page apply button is clicked.
`cancel` ()	Signal emitted when the cancel button is clicked.

---

enum GtkAssistantPageType

An enum for determining the page role inside the GtkAssistant. It's used to handle buttons sensitivity and visibility.

Note that an assistant needs to end its page flow with a page of type `GTK_ASSISTANT_PAGE_CONFIRM`, `GTK_ASSISTANT_PAGE_SUMMARY` or `GTK_ASSISTANT_PAGE_PROGRESS` to be correct.

The Cancel button will only be shown if the page isn't "committed". See `gtk_assistant_commit()` for details.

Members

GTK_ASSISTANT_PAGE_CONTENT	The page has regular contents. Both the Back and forward buttons will be shown.
GTK_ASSISTANT_PAGE_INTRO	The page contains an introduction to the assistant task. Only the Forward button will be shown if there is a next page.
GTK_ASSISTANT_PAGE_CONFIRM	The page lets the user confirm or deny the changes. The Back and Apply buttons will be shown.
GTK_ASSISTANT_PAGE_SUMMARY	The page informs the user of the changes done. Only the Close button will be shown.
GTK_ASSISTANT_PAGE_PROGRESS	Used for tasks that take a long time to complete, blocks the assistant

	until the page is marked as complete. Only the back button will be shown.
GTK_ASSISTANT_PAGE_CUSTOM	Used for when other page types are not appropriate. No buttons will be shown, and the application must add its own buttons through `gtk_assistant_add_action_widget()`.

Property Details

The `“use-header-bar”` property

`“use-header-bar”` `gint`

`TRUE` if the assistant uses a GtkHeaderBar for action buttons instead of the action-area.

For technical reasons, this property is declared as an integer property, but you should only set it to `TRUE` or `FALSE`.

Flags: Read / Write / Construct Only

Allowed values: [-1,1]

Default value: -1

Since: 3.12

Child Property Details

The `“complete”` child property

`“complete”` `gboolean`

Setting the "complete" child property to `TRUE` marks a page as complete (i.e.: all the required fields are filled out). GTK+ uses this information to control the sensitivity of the navigation buttons.

Flags: Read / Write

Default value: FALSE

Since: 2.10

The `“has-padding”` child property

`“has-padding”` `gboolean`

Whether the assistant adds padding around the page.

Flags: Read / Write

Default value: TRUE

The “header-image” child property

“header-image” GdkPixbuf *

This image used to be displayed in the page header.

GtkAssistant:header-image has been deprecated since version 3.2 and should not be used in newly-written code.

Since GTK+ 3.2, a header is no longer shown; add your header decoration to the page content instead.

Flags: Read / Write

Since: 2.10

The “page-type” child property

“page-type” GtkAssistantPageType

The type of the assistant page.

Flags: Read / Write

Default value: GTK_ASSISTANT_PAGE_CONTENT

Since: 2.10

The “sidebar-image” child property

“sidebar-image” GdkPixbuf *

This image used to be displayed in the 'sidebar'.

GtkAssistant:sidebar-image has been deprecated since version 3.2 and should not be used in newly-written code.

Since GTK+ 3.2, the sidebar image is no longer shown.

Flags: Read / Write

Since: 2.10

The “title” child property

“title” gchar *

The title of the page.

Flags: Read / Write

Default value: NULL

Since: 2.10

Style Property Details

The `"content-padding"` style property

`"content-padding"` `gint`

Number of pixels around the content.

`GtkAssistant:content-padding` has been deprecated since version 3.20 and should not be used in newly-written code.

This style property is ignored.

Flags: Read

Allowed values: >= 0

Default value: 1

The `"header-padding"` style property

`"header-padding"` `gint`

Number of pixels around the header.

`GtkAssistant:header-padding` has been deprecated since version 3.20 and should not be used in newly-written code.

This style property is ignored.

Flags: Read

Allowed values: >= 0

Default value: 6

Signal Details

The “apply” signal

```
void
user_function (GtkAssistant *assistant,
               gpointer      user_data)
```

The ::apply signal is emitted when the apply button is clicked.

The default behavior of the GtkAssistant is to switch to the page after the current page, unless the current page is the last one.

A handler for the ::apply signal should carry out the actions for which the wizard has collected data. If the action takes a long time to complete, you might consider putting a page of type GTK_ASSISTANT_PAGE_PROGRESS after the confirmation page and handle this operation within the “prepare” signal of the progress page.

Parameters

assistant	the GtkAssistant
user_data	user data set when the signal handler was connected.

Flags: Run Last

Since: 2.10

The “cancel” signal

```
void
user_function (GtkAssistant *assistant,
               gpointer      user_data)
```

The ::cancel signal is emitted when then the cancel button is clicked.

Parameters

assistant	the GtkAssistant
user_data	user data set when the signal handler was connected.

Flags: Run Last

Since: 2.10

The “close” signal

```
void
user_function (GtkAssistant *assistant,
               gpointer      user_data)
```

The ::close signal is emitted either when the close button of a summary page is clicked, or when the apply button in the last page in the flow (of type GTK_ASSISTANT_PAGE_CONFIRM) is clicked.

Parameters

assistant	the GtkAssistant
user_data	user data set when the signal handler was connected.

Flags: Run Last

Since: 2.10

The “escape” signal

```
void
user_function (GtkAssistant *assistant,
               gpointer      user_data)
```

Flags: Action

The “prepare” signal

```
void
user_function (GtkAssistant *assistant,
               GtkWidget    *page,
               gpointer      user_data)
```

The ::prepare signal is emitted when a new page is set as the assistant's current page, before making the new page visible.

A handler for this signal can do any preparations which are necessary before showing `page` .

Parameters

assistant	the GtkAssistant
page	the current page
user_data	user data set when the signal handler was connected.

Flags: Run Last

Since: 2.10

GtkInvisible

GtkInvisible — A widget which is not displayed

Functions

GtkWidget *	gtk_invisible_new ()
GtkWidget *	gtk_invisible_new_for_screen ()
void	gtk_invisible_set_screen ()
GdkScreen *	gtk_invisible_get_screen ()

Properties

GdkScreen *	screen	Read / Write

Types and Values

struct	GtkInvisible

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkInvisible
```

Implemented Interfaces

GtkInvisible implements AtkImplementorIface and GtkBuildable.

Includes

```
#include <gtk/gtk.h>
```

Description

The GtkInvisible widget is used internally in GTK+, and is probably not very useful for application developers. It is used for reliable pointer grabs and selection handling in the code for drag-and-drop.

Functions

gtk_invisible_new ()

```
GtkWidget *
gtk_invisible_new (void);
```

Creates a new GtkInvisible.

Returns

a new GtkInvisible.

---

gtk_invisible_new_for_screen ()

```
GtkWidget *
gtk_invisible_new_for_screen (GdkScreen *screen);
```

Creates a new GtkInvisible object for a specified screen

Parameters

screen	a GdkScreen which identifies on which the new GtkInvisible will be created.

Returns

a newly created GtkInvisible object

Since: 2.2

---

gtk_invisible_set_screen ()

```
void
gtk_invisible_set_screen (GtkInvisible *invisible,
                          GdkScreen *screen);
```

Sets the GdkScreen where the GtkInvisible object will be displayed.

Parameters

invisible	a GtkInvisible.
screen	a GdkScreen.

Since: 2.2

---

gtk_invisible_get_screen ()

```
GdkScreen *
gtk_invisible_get_screen (GtkInvisible *invisible);
```

Returns the GdkScreen object associated with `invisible`

Parameters

invisible	a GtkInvisible.

Returns

the associated GdkScreen.

[transfer none]

Since: 2.2

Types and Values

struct GtkInvisible

```
struct GtkInvisible;
```

Property Details

The “screen” property

```
  “screen”                   GdkScreen *
```

The screen where this window will be displayed.

Flags: Read / Write

GtkOffscreenWindow

GtkOffscreenWindow — A toplevel to manage offscreen rendering of child widgets

Functions

GtkWidget *	gtk_offscreen_window_new ()
cairo_surface_t *	gtk_offscreen_window_get_surface ()
GdkPixbuf *	gtk_offscreen_window_get_pixbuf ()

Types and Values

struct	GtkOffscreenWindow
struct	GtkOffscreenWindowClass

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkContainer
            └── GtkBin
                └── GtkWindow
                    └── GtkOffscreenWindow
```

Implemented Interfaces

GtkOffscreenWindow implements AtkImplementorIface and GtkBuildable.

Includes

```
#include <gtk/gtk.h>
```

Description

GtkOffscreenWindow is strictly intended to be used for obtaining snapshots of widgets that are not part of a normal widget hierarchy. Since GtkOffscreenWindow is a toplevel widget you cannot obtain snapshots of a full window with it since you cannot pack a toplevel widget in another toplevel.

The idea is to take a widget and manually set the state of it, add it to a GtkOffscreenWindow and then retrieve the snapshot as a cairo_surface_t or GdkPixbuf.

GtkOffscreenWindow derives from GtkWindow only as an implementation detail. Applications should not use any API specific to GtkWindow to operate on this object. It should be treated as a GtkBin that has no parent widget.

When contained offscreen widgets are redrawn, GtkOffscreenWindow will emit a “damage-event” signal.

Functions

gtk_offscreen_window_new ()

```
GtkWidget *
gtk_offscreen_window_new (void);
```

Creates a toplevel container widget that is used to retrieve snapshots of widgets without showing them on the screen.

Returns

A pointer to a GtkWidget

Since: 2.20

---

gtk_offscreen_window_get_surface ()

```
cairo_surface_t *
gtk_offscreen_window_get_surface (GtkOffscreenWindow *offscreen);
```

Retrieves a snapshot of the contained widget in the form of a cairo_surface_t. If you need to keep this around over window resizes then you should add a reference to it.

Parameters

offscreen	the GtkOffscreenWindow contained widget.

Returns

A cairo_surface_t pointer to the offscreen surface, or `NULL`.

[nullable][transfer none]

Since: 2.20

---

gtk_offscreen_window_get_pixbuf ()

```
GdkPixbuf *
gtk_offscreen_window_get_pixbuf (GtkOffscreenWindow *offscreen);
```

Retrieves a snapshot of the contained widget in the form of a GdkPixbuf. This is a new pixbuf with a reference count of 1, and the application should unreference it once it is no longer needed.

Parameters

offscreen	the GtkOffscreenWindow contained widget.

Returns

A GdkPixbuf pointer, or `NULL`.

[nullable][transfer full]

Since: 2.20

Types and Values

struct GtkOffscreenWindow

```
struct GtkOffscreenWindow;
```

---

struct GtkOffscreenWindowClass

```
struct GtkOffscreenWindowClass {
  GtkWindowClass parent_class;
};
```

Members

GtkWindowGroup

GtkWindowGroup — Limit the effect of grabs

Functions

GtkWindowGroup *	gtk_window_group_new ()
void	gtk_window_group_add_window ()
void	gtk_window_group_remove_window ()
GList *	gtk_window_group_list_windows ()
GtkWidget *	gtk_window_group_get_current_grab ()
GtkWidget *	gtk_window_group_get_current_device_grab ()

Types and Values

GtkWindowGroup

Object Hierarchy

```
GObject
└── GtkWindowGroup
```

Includes

```
#include <gtk/gtk.h>
```

Description

A GtkWindowGroup restricts the effect of grabs to windows in the same group, thereby making window groups almost behave like separate applications.

A window can be a member in at most one window group at a time. Windows that have not been explicitly assigned to a group are implicitly treated like windows of the default window group.

GtkWindowGroup objects are referenced by each window in the group, so once you have added all windows to a GtkWindowGroup, you can drop the initial reference to the window group with `g_object_unref()`. If the windows in the window group are subsequently destroyed, then they will be removed from the window group and drop their references on the window group; when all window have been removed, the window group will be freed.

Functions

gtk_window_group_new ()

```
GtkWindowGroup *
gtk_window_group_new (void);
```

Creates a new GtkWindowGroup object. Grabs added with `gtk_grab_add()` only affect windows within the same GtkWindowGroup.

Returns

a new GtkWindowGroup.

---

gtk_window_group_add_window ()

```
void
gtk_window_group_add_window (GtkWindowGroup *window_group,
                             GtkWindow *window);
```

Adds a window to a GtkWindowGroup.

Parameters

window_group	a GtkWindowGroup
window	the GtkWindow to add

---

gtk_window_group_remove_window ()

```
void
gtk_window_group_remove_window (GtkWindowGroup *window_group,
                                GtkWindow *window);
```

Removes a window from a GtkWindowGroup.

Parameters

window_group	a GtkWindowGroup
window	the GtkWindow to remove

---

gtk_window_group_list_windows ()

```
GList *
gtk_window_group_list_windows (GtkWindowGroup *window_group);
```

Returns a list of the GtkWindows that belong to `window_group` .

Parameters

window_group	a GtkWindowGroup

Returns

A newly-allocated list of windows inside the group.

[element-type GtkWindow][transfer container]

Since: 2.14

---

gtk_window_group_get_current_grab ()

```
GtkWidget *
gtk_window_group_get_current_grab (GtkWindowGroup *window_group);
```

Gets the current grab widget of the given group, see `gtk_grab_add()`.

Parameters

window_group	a GtkWindowGroup

Returns

the current grab widget of the group.

[transfer none]

Since: 2.22

---

gtk_window_group_get_current_device_grab ()

```
GtkWidget *
gtk_window_group_get_current_device_grab
                                (GtkWindowGroup *window_group,
                                 GdkDevice *device);
```

Returns the current grab widget for `device` , or `NULL` if none.

Parameters

window_group	a GtkWindowGroup
device	a GdkDevice

Returns

The grab widget, or `NULL`.

[nullable][transfer none]

Since: 3.0

Types and Values

GtkWindowGroup

```
typedef struct _GtkWindowGroup GtkWindowGroup;
```

Layout Containers

GtkBox — A container for packing widgets in a single row or column

GtkGrid — Pack widgets in rows and columns

GtkRevealer — Hide and show with animation

GtkListBox — A list container

GtkFlowBox — A container that allows reflowing its children

GtkStack — A stacking container

GtkStackSwitcher — A controller for GtkStack

GtkStackSidebar — An automatic sidebar widget

GtkActionBar — A full width bar for presenting contextual actions

GtkHeaderBar — A box with a centered child

GtkOverlay — A container which overlays widgets on top of each other

GtkButtonBox — A container for arranging buttons

GtkPaned — A widget with two adjustable panes

GtkLayout — Infinite scrollable area containing child widgets and/or custom drawing

GtkNotebook — A tabbed notebook container

GtkExpander — A container which can hide its child

GtkOrientable — An interface for flippable widgets

GtkAspectFrame — A frame that constrains its child to a particular aspect ratio

GtkFixed — A container which allows you to position widgets at fixed coordinates

---

GtkBox

GtkBox — A container for packing widgets in a single row or column

Functions

GtkWidget *	gtk_box_new ()
void	gtk_box_pack_start ()
void	gtk_box_pack_end ()
gboolean	gtk_box_get_homogeneous ()
void	gtk_box_set_homogeneous ()
gint	gtk_box_get_spacing ()
void	gtk_box_set_spacing ()
void	gtk_box_reorder_child ()
void	gtk_box_query_child_packing ()
void	gtk_box_set_child_packing ()
GtkBaselinePosition	gtk_box_get_baseline_position ()
void	gtk_box_set_baseline_position ()
GtkWidget *	gtk_box_get_center_widget ()
void	gtk_box_set_center_widget ()

Properties

GtkBaselinePosition	baseline-position	Read / Write
gboolean	homogeneous	Read / Write
gint	spacing	Read / Write

Child Properties

gboolean	expand	Read / Write
gboolean	fill	Read / Write
GtkPackType	pack-type	Read / Write
guint	padding	Read / Write
gint	position	Read / Write

Types and Values

struct	GtkBox
struct	GtkBoxClass

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkContainer
            └── GtkBox
                ├── GtkAppChooserWidget
                ├── GtkButtonBox
                ├── GtkColorChooserWidget
                ├── GtkColorSelection
                ├── GtkFileChooserButton
                ├── GtkFileChooserWidget
                ├── GtkFontChooserWidget
                ├── GtkFontSelection
                ├── GtkHBox
                ├── GtkInfoBar
                ├── GtkRecentChooserWidget
                ├── GtkShortcutsSection
                ├── GtkShortcutsGroup
                ├── GtkShortcutsShortcut
                ├── GtkStackSwitcher
                ├── GtkStatusbar
                └── GtkVBox
```

Implemented Interfaces

GtkBox implements AtkImplementorIface, GtkBuildable and GtkOrientable.

Includes

```
#include <gtk/gtk.h>
```

Description

The GtkBox widget arranges child widgets into a single row or column, depending upon the value of its “orientation” property. Within the other dimension, all children are allocated the same size. Of course, the “halign” and “valign” properties can be used on the children to influence their allocation.

GtkBox uses a notion of packing. Packing refers to adding widgets with reference to a particular position in a GtkContainer. For a GtkBox, there are two reference positions: the start and the end of the box. For a vertical GtkBox, the start is defined as the top of the box and the end is defined as the bottom. For a horizontal GtkBox the start is defined as the left side and the end is defined as the right side.

Use repeated calls to `gtk_box_pack_start()` to pack widgets into a GtkBox from start to end. Use `gtk_box_pack_end()` to add widgets from end to start. You may intersperse these calls and add widgets from both ends of the same GtkBox.

Because GtkBox is a GtkContainer, you may also use `gtk_container_add()` to insert widgets into the box, and they will be packed with the default values for expand and fill child properties. Use `gtk_container_remove()` to remove widgets from the GtkBox.

Use `gtk_box_set_homogeneous()` to specify whether or not all children of the GtkBox are forced to get the same amount of space.

Use gtk_box_set_spacing() to determine how much space will be minimally placed between all children in the GtkBox. Note that spacing is added between the children, while padding added by gtk_box_pack_start() or gtk_box_pack_end() is added on either side of the widget it belongs to.

Use gtk_box_reorder_child() to move a GtkBox child to a different place in the box.

Use gtk_box_set_child_packing() to reset the expand, fill and padding child properties. Use gtk_box_query_child_packing() to query these fields.

CSS nodes

GtkBox uses a single CSS node with name box.

In horizontal orientation, the nodes of the children are always arranged from left to right. So :first-child will always select the leftmost child, regardless of text direction.

Functions

gtk_box_new ()

```
GtkWidget *
gtk_box_new (GtkOrientation orientation,
             gint spacing);
```

Creates a new GtkBox.

Parameters

orientation	the box’s orientation.
spacing	the number of pixels to place by default between children.

Returns

a new GtkBox.

Since: 3.0

gtk_box_pack_start ()

```
void
gtk_box_pack_start (GtkBox *box,
                    GtkWidget *child,
                    gboolean expand,
                    gboolean fill,
                    guint padding);
```

Adds `child` to `box` , packed with reference to the start of `box` . The `child` is packed after any other child packed with reference to the start of `box` .

Parameters

box	a GtkBox
child	the GtkWidget to be added to `box`
expand	`TRUE` if the new child is to be given extra space allocated to `box` . The extra space will be divided evenly between all children that use this option
fill	`TRUE` if space given to `child` by the `expand` option is actually allocated to `child` , rather than just padding it. This parameter has no effect if `expand` is set to `FALSE`. A child is always allocated the full height of a horizontal GtkBox and the full width of a vertical GtkBox. This option affects the other dimension
padding	extra space in pixels to put between this child and its neighbors, over and above the global amount specified by "spacing" property. If `child` is a widget at one of the reference ends of `box` , then `padding` pixels are also put between `child` and the reference edge of `box`

---

gtk_box_pack_end ()

```
void
gtk_box_pack_end (GtkBox *box,
                  GtkWidget *child,
                  gboolean expand,
                  gboolean fill,
                  guint padding);
```

Adds `child` to `box` , packed with reference to the end of `box` . The `child` is packed after (away from end of) any other child packed with reference to the end of `box` .

Parameters

box	a GtkBox
child	the GtkWidget to be added to `box`
expand	`TRUE` if the new child is to be given extra space allocated to `box` . The extra space will be divided evenly between all children of `box` that use this option
fill	`TRUE` if space given to `child` by the `expand` option is actually allocated to `child` , rather than just padding it. This parameter has no effect if `expand` is set to `FALSE`. A child is always allocated the full height of a horizontal GtkBox and the full width of a vertical GtkBox. This option affects the other dimension
padding	extra space in pixels to put between this child and its neighbors, over and above the global amount specified by "spacing" property. If `child` is a widget at one of the reference ends of `box` , then `padding` pixels are also put between `child` and the reference edge of `box`

---

gtk_box_get_homogeneous ()

```
gboolean
gtk_box_get_homogeneous (GtkBox *box);
```

Returns whether the box is homogeneous (all children are the same size). See `gtk_box_set_homogeneous()`.

Parameters

box	a GtkBox

Returns

`TRUE` if the box is homogeneous.

---

gtk_box_set_homogeneous ()

```
void
gtk_box_set_homogeneous (GtkBox *box,
                         gboolean homogeneous);
```

Sets the "homogeneous" property of `box` , controlling whether or not all children of `box` are given equal space in the box.

Parameters

box	a GtkBox
homogeneous	a boolean value, `TRUE` to create equal allotments, `FALSE` for variable allotments

---

gtk_box_get_spacing ()

```
gint
gtk_box_get_spacing (GtkBox *box);
```

Gets the value set by `gtk_box_set_spacing()`.

Parameters

box	a GtkBox

Returns

spacing between children

---

gtk_box_set_spacing ()

```
void
gtk_box_set_spacing (GtkBox *box,
                     gint spacing);
```

Sets the "spacing" property of `box` , which is the number of pixels to place between children of `box` .

Parameters

box	a GtkBox
spacing	the number of pixels to put between children

---

gtk_box_reorder_child ()

```
void
gtk_box_reorder_child (GtkBox *box,
                       GtkWidget *child,
                       gint position);
```

Moves `child` to a new `position` in the list of `box` children. The list contains widgets packed GTK_PACK_START as well as widgets packed GTK_PACK_END, in the order that these widgets were added to `box` .

A widget's position in the `box` children list determines where the widget is packed into `box` . A child widget at some position in the list will be packed just after all other widgets of the same packing type that appear earlier in the list.

Parameters

box	a GtkBox
child	the GtkWidget to move
position	the new position for `child` in the list of children of `box` , starting from 0. If negative, indicates the end of the list

---

gtk_box_query_child_packing ()

```
void
gtk_box_query_child_packing (GtkBox *box,
                             GtkWidget *child,
                             gboolean *expand,
                             gboolean *fill,
                             guint *padding,
                             GtkPackType *pack_type);
```

Obtains information about how `child` is packed into `box` .

Parameters

box	a GtkBox	
child	the GtkWidget of the child to query	
expand	pointer to return location for expand child property.	[out]
fill	pointer to return location for fill child property.	[out]
padding	pointer to return location for padding child property.	[out]
pack_type	pointer to return location for pack-type child property.	[out]

---

gtk_box_set_child_packing ()

```
void
gtk_box_set_child_packing (GtkBox *box,
                           GtkWidget *child,
                           gboolean expand,
                           gboolean fill,
                           guint padding,
                           GtkPackType pack_type);
```

Sets the way `child` is packed into `box` .

Parameters

box	a GtkBox
child	the GtkWidget of the child to set
expand	the new value of the expand child property
fill	the new value of the fill child property
padding	the new value of the padding child property
pack_type	the new value of the pack-type child property

---

gtk_box_get_baseline_position ()

```
GtkBaselinePosition
gtk_box_get_baseline_position (GtkBox *box);
```

Gets the value set by `gtk_box_set_baseline_position()`.

Parameters

box	a GtkBox

Returns

the baseline position

Since: 3.10

---

gtk_box_set_baseline_position ()

```
void
gtk_box_set_baseline_position (GtkBox *box,
                               GtkBaselinePosition position);
```

Sets the baseline position of a box. This affects only horizontal boxes with at least one baseline aligned child. If there is more vertical space available than requested, and the baseline is not allocated by the parent then `position` is used to allocate the baseline wrt the extra space available.

Parameters

box	a GtkBox	
position	a GtkBaselinePosition	

Since: 3.10

---

gtk_box_get_center_widget ()

```
GtkWidget *
gtk_box_get_center_widget (GtkBox *box);
```

Retrieves the center widget of the box.

Parameters

box	a GtkBox	

Returns

the center widget or `NULL` in case no center widget is set.

[transfer none][nullable]

Since: 3.12

---

gtk_box_set_center_widget ()

```
void
gtk_box_set_center_widget (GtkBox *box,
                           GtkWidget *widget);
```

Sets a center widget; that is a child widget that will be centered with respect to the full width of the box, even if the children at either side take up different amounts of space.

Parameters

box	a GtkBox	
widget	the widget to center.	[allow-none]

Since: 3.12

Types and Values

struct GtkBox

```
struct GtkBox;
```

---

struct GtkBoxClass

```
struct GtkBoxClass {
  GtkContainerClass parent_class;
};
```

Members

Property Details

The “baseline-position” property

  “baseline-position” GtkBaselinePosition

The position of the baseline aligned widgets if extra space is available.

Flags: Read / Write

Default value: GTK_BASELINE_POSITION_CENTER

The “homogeneous” property

  “homogeneous” gboolean

Whether the children should all be the same size.

Flags: Read / Write

Default value: FALSE

The “spacing” property

  “spacing” gint

The amount of space between children.

Flags: Read / Write

Allowed values: >= 0

Default value: 0

Child Property Details

The “expand” child property

  “expand” gboolean

Whether the child should receive extra space when the parent grows.

Note that the default value for this property is FALSE for GtkBox, but GtkHBox, GtkVBox and other subclasses

use the old default of TRUE.

Note: The “hexpand” or “vexpand” properties are the preferred way to influence whether the child receives extra space, by setting the child’s expand property corresponding to the box’s orientation.

In contrast to “hexpand”, the expand child property does not cause the box to expand itself.

Flags: Read / Write

Default value: FALSE

The “fill” child property

“fill” gboolean

Whether the child should fill extra space or use it as padding.

Note: The “halign” or “valign” properties are the preferred way to influence whether the child fills available space, by setting the child’s align property corresponding to the box’s orientation to GTK_ALIGN_FILL to fill, or to something else to refrain from filling.

Flags: Read / Write

Default value: TRUE

The “pack-type” child property

“pack-type” GtkPackType

A GtkPackType indicating whether the child is packed with reference to the start or end of the parent.

Flags: Read / Write

Default value: GTK_PACK_START

The “padding” child property

“padding” guint

Extra space to put between the child and its neighbors, in pixels.

Note: The CSS padding properties are the preferred way to add space among widgets, by setting the paddings corresponding to the box’s orientation.

Flags: Read / Write

Allowed values: <= G_MAXINT

Default value: 0

The “position” child property

“position” gint

The index of the child in the parent.

Flags: Read / Write

Allowed values: >= -1

Default value: 0

See Also

GtkGrid

GtkGrid

GtkGrid — Pack widgets in rows and columns

Functions

GtkWidget *	gtk_grid_new ()
void	gtk_grid_attach ()
void	gtk_grid_attach_next_to ()
GtkWidget *	gtk_grid_get_child_at ()
void	gtk_grid_insert_row ()
void	gtk_grid_insert_column ()
void	gtk_grid_remove_row ()
void	gtk_grid_remove_column ()
void	gtk_grid_insert_next_to ()
void	gtk_grid_set_row_homogeneous ()
gboolean	gtk_grid_get_row_homogeneous ()
void	gtk_grid_set_row_spacing ()
guint	gtk_grid_get_row_spacing ()
void	gtk_grid_set_column_homogeneous ()
gboolean	gtk_grid_get_column_homogeneous ()
void	gtk_grid_set_column_spacing ()
guint	gtk_grid_get_column_spacing ()
gint	gtk_grid_get_baseline_row ()
void	gtk_grid_set_baseline_row ()
GtkBaselinePosition	gtk_grid_get_row_baseline_position ()
void	gtk_grid_set_row_baseline_position ()

Properties

gint	baseline-row	Read / Write
gboolean	column-homogeneous	Read / Write
gint	column-spacing	Read / Write
gboolean	row-homogeneous	Read / Write

gint	row-spacing	Read / Write

Child Properties

gint	height	Read / Write
gint	left-attach	Read / Write
gint	top-attach	Read / Write
gint	width	Read / Write

Types and Values

struct	GtkGrid
struct	GtkGridClass

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkContainer
            └── GtkGrid
```

Implemented Interfaces

GtkGrid implements AtkImplementorIface, GtkBuildable and GtkOrientable.

Includes

```
#include <gtk/gtk.h>
```

Description

GtkGrid is a container which arranges its child widgets in rows and columns, with arbitrary positions and horizontal/vertical spans.

Children are added using `gtk_grid_attach()`. They can span multiple rows or columns. It is also possible to add a child next to an existing child, using `gtk_grid_attach_next_to()`. The behaviour of GtkGrid when several children occupy the same grid cell is undefined.

GtkGrid can be used like a GtkBox by just using `gtk_container_add()`, which will place children next to each other in the direction determined by the "orientation" property. However, if all you want is a single row or column, then GtkBox is the preferred widget.

CSS nodes

GtkGrid uses a single CSS node with name grid.

Functions

gtk_grid_new ()

```
GtkWidget *
gtk_grid_new (void);
```

Creates a new grid widget.

Returns

the new GtkGrid

---

gtk_grid_attach ()

```
void
gtk_grid_attach (GtkGrid *grid,
                 GtkWidget *child,
                 gint left,
                 gint top,
                 gint width,
                 gint height);
```

Adds a widget to the grid.

The position of `child` is determined by `left` and `top` . The number of “cells” that `child` will occupy is determined by `width` and `height` .

Parameters

grid	a GtkGrid
child	the widget to add
left	the column number to attach the left side of `child` to
top	the row number to attach the top side of `child` to
width	the number of columns that `child` will span
height	the number of rows that `child` will span

---

gtk_grid_attach_next_to ()

```
void
gtk_grid_attach_next_to (GtkGrid *grid,
                         GtkWidget *child,
                         GtkWidget *sibling,
                         GtkPositionType side,
                         gint width,
                         gint height);
```

Adds a widget to the grid.

The widget is placed next to `sibling` , on the side determined by `side` . When `sibling` is `NULL`, the widget is placed in row (for left or right placement) or column 0 (for top or bottom placement), at the end indicated by `side` .

Attaching widgets labeled [1], [2], [3] with `sibling` == `NULL` and `side` == `GTK_POS_LEFT` yields a layout of 3[1].

Parameters

grid	a GtkGrid	
child	the widget to add	
sibling	the child of `grid` that `child` will be placed next to, or `NULL` to place `child` at the beginning or end.	[allow-none]
side	the side of `sibling` that `child` is positioned next to	
width	the number of columns that `child` will span	
height	the number of rows that `child` will span	

---

gtk_grid_get_child_at ()

```
GtkWidget *
gtk_grid_get_child_at (GtkGrid *grid,
                       gint left,
                       gint top);
```

Gets the child of `grid` whose area covers the grid cell whose upper left corner is at `left` , `top` .

Parameters

grid	a GtkGrid
left	the left edge of the cell
top	the top edge of the cell

Returns

the child at the given position, or `NULL`.

[transfer none][nullable]

Since: 3.2

---

gtk_grid_insert_row ()

```
void
gtk_grid_insert_row (GtkGrid *grid,
                     gint position);
```

Inserts a row at the specified position.

Children which are attached at or below this position are moved one row down. Children which span across this position are grown to span the new row.

Parameters

grid	a GtkGrid
position	the position to insert the row at

Since: 3.2

---

gtk_grid_insert_column ()

```
void
gtk_grid_insert_column (GtkGrid *grid,
                        gint position);
```

Inserts a column at the specified position.

Children which are attached at or to the right of this position are moved one column to the right. Children which span across this position are grown to span the new column.

Parameters

grid	a GtkGrid
position	the position to insert the column at

Since: 3.2

---

gtk_grid_remove_row ()

```
void
gtk_grid_remove_row (GtkGrid *grid,
                     gint position);
```

Removes a row from the grid.

Children that are placed in this row are removed, spanning children that overlap this row have their height reduced by one, and children below the row are moved up.

Parameters

grid	a GtkGrid
position	the position of the row to remove

Since: 3.10

---

gtk_grid_remove_column ()

```
void
gtk_grid_remove_column (GtkGrid *grid,
                        gint position);
```

Removes a column from the grid.

Children that are placed in this column are removed, spanning children that overlap this column have their width reduced by one, and children after the column are moved to the left.

Parameters

grid	a GtkGrid
position	the position of the column to remove

Since: 3.10

---

gtk_grid_insert_next_to ()

```
void
gtk_grid_insert_next_to (GtkGrid *grid,
                         GtkWidget *sibling,
                         GtkPositionType side);
```

Inserts a row or column at the specified position.

The new row or column is placed next to `sibling` , on the side determined by `side` . If `side` is `GTK_POS_TOP` or `GTK_POS_BOTTOM`, a row is inserted. If `side` is `GTK_POS_LEFT` of `GTK_POS_RIGHT`, a column is inserted.

Parameters

grid	a GtkGrid
sibling	the child of `grid` that the new row or column will be placed next to
side	the side of `sibling` that `child` is positioned next to

Since: 3.2

---

gtk_grid_set_row_homogeneous ()

```
void
gtk_grid_set_row_homogeneous (GtkGrid *grid,
                              gboolean homogeneous);
```

Sets whether all rows of `grid` will have the same height.

Parameters

grid	a GtkGrid
homogeneous	TRUE to make rows homogeneous

---

gtk_grid_get_row_homogeneous ()

```
gboolean
gtk_grid_get_row_homogeneous (GtkGrid *grid);
```

Returns whether all rows of grid have the same height.

Parameters

grid	a GtkGrid

Returns

whether all rows of grid have the same height.

---

gtk_grid_set_row_spacing ()

```
void
gtk_grid_set_row_spacing (GtkGrid *grid,
                          guint spacing);
```

Sets the amount of space between rows of grid .

Parameters

grid	a GtkGrid
spacing	the amount of space to insert between rows

---

gtk_grid_get_row_spacing ()

```
guint
gtk_grid_get_row_spacing (GtkGrid *grid);
```

Returns the amount of space between the rows of grid .

Parameters

grid	a GtkGrid

Returns

the row spacing of `grid`

---

gtk_grid_set_column_homogeneous ()

```
void
gtk_grid_set_column_homogeneous (GtkGrid *grid,
                                 gboolean homogeneous);
```

Sets whether all columns of `grid` will have the same width.

Parameters

grid	a GtkGrid
homogeneous	`TRUE` to make columns homogeneous

---

gtk_grid_get_column_homogeneous ()

```
gboolean
gtk_grid_get_column_homogeneous (GtkGrid *grid);
```

Returns whether all columns of `grid` have the same width.

Parameters

grid	a GtkGrid

Returns

whether all columns of `grid` have the same width.

---

gtk_grid_set_column_spacing ()

```
void
gtk_grid_set_column_spacing (GtkGrid *grid,
                             guint spacing);
```

Sets the amount of space between columns of `grid` .

Parameters

grid	a GtkGrid
spacing	the amount of space to insert between columns

---

gtk_grid_get_column_spacing ()

```
guint
gtk_grid_get_column_spacing (GtkGrid *grid);
```

Returns the amount of space between the columns of `grid` .

Parameters

grid	a GtkGrid

Returns

the column spacing of `grid`

---

gtk_grid_get_baseline_row ()

```
gint
gtk_grid_get_baseline_row (GtkGrid *grid);
```

Returns which row defines the global baseline of `grid` .

Parameters

grid	a GtkGrid

Returns

the row index defining the global baseline

Since: 3.10

---

gtk_grid_set_baseline_row ()

```
void
gtk_grid_set_baseline_row (GtkGrid *grid,
                           gint row);
```

Sets which row defines the global baseline for the entire grid. Each row in the grid can have its own local baseline, but only one of those is global, meaning it will be the baseline in the parent of the `grid` .

Parameters

grid	a GtkGrid
row	the row index

Since: 3.10

---

gtk_grid_get_row_baseline_position ()

```
GtkBaselinePosition
gtk_grid_get_row_baseline_position (GtkGrid *grid,
                                    gint row);
```

Returns the baseline position of `row` as set by `gtk_grid_set_row_baseline_position()` or the default value `GTK_BASELINE_POSITION_CENTER`.

Parameters

grid	a GtkGrid
row	a row index

Returns

the baseline position of `row`

Since: 3.10

---

gtk_grid_set_row_baseline_position ()

```
void
gtk_grid_set_row_baseline_position (GtkGrid *grid,
                                    gint row,
                                    GtkBaselinePosition pos);
```

Sets how the baseline should be positioned on `row` of the grid, in case that row is assigned more space than is requested.

Parameters

grid	a GtkGrid
row	a row index
pos	a GtkBaselinePosition

Since: 3.10

Types and Values

struct GtkGrid

```
struct GtkGrid;
```

---

struct GtkGridClass

```
struct GtkGridClass {
  GtkContainerClass parent_class;
};
```

Members

Property Details

The “baseline-row” property

  “baseline-row” gint

The row to align the to the baseline when valign is GTK_ALIGN_BASELINE.

Flags: Read / Write

Allowed values: >= 0

Default value: 0

The “column-homogeneous” property

  “column-homogeneous” gboolean

If TRUE, the columns are all the same width.

Flags: Read / Write

Default value: FALSE

The “column-spacing” property

  “column-spacing” gint

The amount of space between two consecutive columns.

Flags: Read / Write

Allowed values: [0,32767]

Default value: 0

The “row-homogeneous” property

  “row-homogeneous” gboolean

If TRUE, the rows are all the same height.

Flags: Read / Write

Default value: FALSE

The “row-spacing” property

“row-spacing” gint

The amount of space between two consecutive rows.

Flags: Read / Write

Allowed values: [0,32767]

Default value: 0

Child Property Details

The “height” child property

“height” gint

The number of rows that a child spans.

Flags: Read / Write

Allowed values: >= 1

Default value: 1

The “left-attach” child property

“left-attach” gint

The column number to attach the left side of the child to.

Flags: Read / Write

Default value: 0

The “top-attach” child property

“top-attach” gint

The row number to attach the top side of a child widget to.

Flags: Read / Write

Default value: 0

The “width” child property

“width” gint

The number of columns that a child spans.

Flags: Read / Write

Allowed values: >= 1

Default value: 1

See Also

GtkBox

GtkRevealer

GtkRevealer — Hide and show with animation

Functions

GtkWidget *	gtk_revealer_new ()
gboolean	gtk_revealer_get_reveal_child ()
void	gtk_revealer_set_reveal_child ()
gboolean	gtk_revealer_get_child_revealed ()
guint	gtk_revealer_get_transition_duration ()
void	gtk_revealer_set_transition_duration ()
GtkRevealerTransitionType	gtk_revealer_get_transition_type ()
void	gtk_revealer_set_transition_type ()

Properties

gboolean	child-revealed	Read
gboolean	reveal-child	Read / Write / Construct
guint	transition-duration	Read / Write / Construct
GtkRevealerTransitionType	transition-type	Read / Write / Construct

Types and Values

struct	GtkRevealer
struct	GtkRevealerClass
enum	GtkRevealerTransitionType

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkContainer
```

```
            └── GtkBin
                └── GtkRevealer
```

Implemented Interfaces

GtkRevealer implements AtkImplementorIface and GtkBuildable.

Includes

```
#include <gtk/gtk.h>
```

Description

The GtkRevealer widget is a container which animates the transition of its child from invisible to visible.

The style of transition can be controlled with `gtk_revealer_set_transition_type()`.

These animations respect the "gtk-enable-animations" setting.

CSS nodes

GtkRevealer has a single CSS node with name revealer.

The GtkRevealer widget was added in GTK+ 3.10.

Functions

gtk_revealer_new ()

```
GtkWidget *
gtk_revealer_new (void);
```

Creates a new GtkRevealer.

Returns

a newly created GtkRevealer

Since: 3.10

---

gtk_revealer_get_reveal_child ()

```
gboolean
gtk_revealer_get_reveal_child (GtkRevealer *revealer);
```

Returns whether the child is currently revealed. See `gtk_revealer_set_reveal_child()`.

This function returns `TRUE` as soon as the transition is to the revealed state is started. To learn whether the child

is fully revealed (ie the transition is completed), use `gtk_revealer_get_child_revealed()`.

Parameters

revealer	a GtkRevealer

Returns

`TRUE` if the child is revealed.

Since: 3.10

---

gtk_revealer_set_reveal_child ()

```
void
gtk_revealer_set_reveal_child (GtkRevealer *revealer,
                               gboolean reveal_child);
```

Tells the GtkRevealer to reveal or conceal its child.

The transition will be animated with the current transition type of `revealer` .

Parameters

revealer	a GtkRevealer
reveal_child	`TRUE` to reveal the child

Since: 3.10

---

gtk_revealer_get_child_revealed ()

```
gboolean
gtk_revealer_get_child_revealed (GtkRevealer *revealer);
```

Returns whether the child is fully revealed, in other words whether the transition to the revealed state is completed.

Parameters

revealer	a GtkRevealer

Returns

`TRUE` if the child is fully revealed

Since: 3.10

---

gtk_revealer_get_transition_duration ()

```
guint
gtk_revealer_get_transition_duration (GtkRevealer *revealer);
```

Returns the amount of time (in milliseconds) that transitions will take.

Parameters

revealer	a GtkRevealer

Returns

the transition duration

Since: 3.10

---

gtk_revealer_set_transition_duration ()

```
void
gtk_revealer_set_transition_duration (GtkRevealer *revealer,
                                      guint duration);
```

Sets the duration that transitions will take.

Parameters

revealer	a GtkRevealer
duration	the new duration, in milliseconds

Since: 3.10

---

gtk_revealer_get_transition_type ()

```
GtkRevealerTransitionType
gtk_revealer_get_transition_type (GtkRevealer *revealer);
```

Gets the type of animation that will be used for transitions in `revealer` .

Parameters

revealer	a GtkRevealer

Returns

the current transition type of `revealer`

Since: 3.10

---

gtk_revealer_set_transition_type ()

```
void
gtk_revealer_set_transition_type (GtkRevealer *revealer,
                                  GtkRevealerTransitionType transition);
```

Sets the type of animation that will be used for transitions in `revealer` . Available types include various kinds of fades and slides.

Parameters

revealer	a GtkRevealer
transition	the new transition type

Since: 3.10

Types and Values

struct GtkRevealer

```
struct GtkRevealer;
```

---

struct GtkRevealerClass

```
struct GtkRevealerClass {
  GtkBinClass parent_class;
};
```

Members

---

enum GtkRevealerTransitionType

These enumeration values describe the possible transitions when the child of a GtkRevealer widget is shown or hidden.

Members

GTK_REVEALER_TRANSITION_TYPE_NONE	No transition
GTK_REVEALER_TRANSITION_TYPE_CROSSFADE	Fade in
GTK_REVEALER_TRANSITION_TYPE_SLIDE_RIGHT	Slide in from the left
GTK_REVEALER_TRANSITION_TYPE_SLIDE_LEFT	Slide in from the right

GTK_REVEALER_TRANSITION_TYPE_SLIDE_UP	Slide in from the bottom
GTK_REVEALER_TRANSITION_TYPE_SLIDE_DOWN	Slide in from the top

Property Details

The `"child-revealed"` property

`"child-revealed"` `gboolean`

Whether the child is revealed and the animation target reached.

Flags: Read

Default value: FALSE

The `"reveal-child"` property

`"reveal-child"` `gboolean`

Whether the container should reveal the child.

Flags: Read / Write / Construct

Default value: FALSE

The `"transition-duration"` property

`"transition-duration"` `guint`

The animation duration, in milliseconds.

Flags: Read / Write / Construct

Default value: 250

The `"transition-type"` property

`"transition-type"` `GtkRevealerTransitionType`

The type of animation used to transition.

Flags: Read / Write / Construct

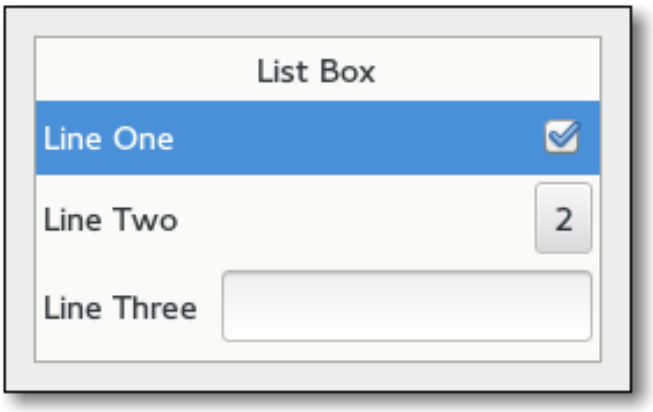
Default value: GTK_REVEALER_TRANSITION_TYPE_SLIDE_DOWN

See Also

GtkExpander

GtkListBox

GtkListBox — A list container



Functions

gboolean	(*GtkListBoxFilterFunc) ()
gint	(*GtkListBoxSortFunc) ()
void	(*GtkListBoxUpdateHeaderFunc) ()
GtkWidget *	gtk_list_box_new ()
void	gtk_list_box_prepend ()
void	gtk_list_box_insert ()
void	gtk_list_box_select_row ()
void	gtk_list_box_unselect_row ()
void	gtk_list_box_select_all ()
void	gtk_list_box_unselect_all ()
GtkListBoxRow *	gtk_list_box_get_selected_row ()
void	(*GtkListBoxForeachFunc) ()
void	gtk_list_box_selected_foreach ()
GList *	gtk_list_box_get_selected_rows ()
void	gtk_list_box_set_selection_mode ()
GtkSelectionMode	gtk_list_box_get_selection_mode ()
void	gtk_list_box_set_activate_on_single_click ()
gboolean	gtk_list_box_get_activate_on_single_click ()
GtkAdjustment *	gtk_list_box_get_adjustment ()
void	gtk_list_box_set_adjustment ()
void	gtk_list_box_set_placeholder ()
GtkListBoxRow *	gtk_list_box_get_row_at_index ()
GtkListBoxRow *	gtk_list_box_get_row_at_y ()
void	gtk_list_box_invalidate_filter ()
void	gtk_list_box_invalidate_headers ()
void	gtk_list_box_invalidate_sort ()
void	gtk_list_box_set_filter_func ()
void	gtk_list_box_set_header_func ()
void	gtk_list_box_set_sort_func ()
void	gtk_list_box_drag_highlight_row ()
void	gtk_list_box_drag_unhighlight_row ()
GtkWidget *	(*GtkListBoxCreateWidgetFunc) ()

void	gtk_list_box_bind_model ()
GtkWidget *	gtk_list_box_row_new ()
void	gtk_list_box_row_changed ()
gboolean	gtk_list_box_row_is_selected ()
GtkWidget *	gtk_list_box_row_get_header ()
void	gtk_list_box_row_set_header ()
gint	gtk_list_box_row_get_index ()
void	gtk_list_box_row_set_activatable ()
gboolean	gtk_list_box_row_get_activatable ()
void	gtk_list_box_row_set_selectable ()
gboolean	gtk_list_box_row_get_selectable ()

Properties

gboolean	activate-on-single-click	Read / Write
GtkSelectionMode	selection-mode	Read / Write
gboolean	activatable	Read / Write
gboolean	selectable	Read / Write

Signals

void	activate-cursor-row	Action
void	move-cursor	Action
void	row-activated	Run Last
void	row-selected	Run Last
void	select-all	Action
void	selected-rows-changed	Run First
void	toggle-cursor-row	Action
void	unselect-all	Action
void	activate	Action

Types and Values

struct	GtkListBox
struct	GtkListBoxClass
struct	GtkListBoxRow
struct	GtkListBoxRowClass

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkContainer
            ├── GtkBin
            │   └── GtkListBoxRow
            └── GtkListBox
```

Implemented Interfaces

GtkListBox implements AtkImplementorIface and GtkBuildable.

GtkListBoxRow implements AtkImplementorIface, GtkBuildable and GtkActionable.

Includes

```
#include <gtk/gtk.h>
```

Description

A GtkListBox is a vertical container that contains GtkListBoxRow children. These rows can by dynamically sorted and filtered, and headers can be added dynamically depending on the row content. It also allows keyboard and mouse navigation and selection like a typical list.

Using GtkListBox is often an alternative to GtkTreeView, especially when the list contents has a more complicated layout than what is allowed by a GtkCellRenderer, or when the contents is interactive (i.e. has a button in it).

Although a GtkListBox must have only GtkListBoxRow children you can add any kind of widget to it via `gtk_container_add()`, and a GtkListBoxRow widget will automatically be inserted between the list and the widget.

GtkListBoxRows can be marked as activatable or selectable. If a row is activatable, "row-activated" will be emitted for it when the user tries to activate it. If it is selectable, the row will be marked as selected when the user tries to select it.

The GtkListBox widget was added in GTK+ 3.10.

GtkListBox as GtkBuildable

The GtkListBox implementation of the GtkBuildable interface supports setting a child as the placeholder by specifying "placeholder" as the "type" attribute of a <child> element. See `gtk_list_box_set_placeholder()` for info.

CSS nodes

```
1                                                  list
2                                                  └── row[.activatable]
```

GtkListBox uses a single CSS node named list. Each GtkListBoxRow uses a single CSS node named row. The row nodes get the .activatable style class added when appropriate.

Functions

GtkListBoxFilterFunc ()

```
gboolean
(*GtkListBoxFilterFunc) (GtkListBoxRow *row,
                         gpointer user_data);
```

Will be called whenever the row changes or is added and lets you control if the row should be visible or not.

Parameters

row	the row that may be filtered	
user_data	user data.	[closure]

Returns

`TRUE` if the row should be visible, `FALSE` otherwise

Since: 3.10

GtkListBoxSortFunc ()

```
gint
(*GtkListBoxSortFunc) (GtkListBoxRow *row1,
                       GtkListBoxRow *row2,
                       gpointer user_data);
```

Compare two rows to determine which should be first.

Parameters

row1	the first row	
row2	the second row	
user_data	user data.	[closure]

Returns

< 0 if `row1` should be before `row2` , 0 if they are equal and > 0 otherwise

Since: 3.10

GtkListBoxUpdateHeaderFunc ()

```
void
(*GtkListBoxUpdateHeaderFunc) (GtkListBoxRow *row,
                               GtkListBoxRow *before,
                               gpointer user_data);
```

Whenever `row` changes or which row is before `row` changes this is called, which lets you update the header on `row` . You may remove or set a new one via `gtk_list_box_row_set_header()` or just change the state of the current header widget.

Parameters

row	the row to update	
before	the row before `row` , or `NULL` if it is first.	[allow-none]
user_data	user data.	[closure]

Since: 3.10

---

gtk_list_box_new ()

```
GtkWidget *
gtk_list_box_new (void);
```

Creates a new GtkListBox container.

Returns

a new GtkListBox

Since: 3.10

---

gtk_list_box_prepend ()

```
void
gtk_list_box_prepend (GtkListBox *box,
                      GtkWidget *child);
```

Prepend a widget to the list. If a sort function is set, the widget will actually be inserted at the calculated position and this function has the same effect of `gtk_container_add()`.

Parameters

box	a GtkListBox
child	the GtkWidget to add

Since: 3.10

---

gtk_list_box_insert ()

```
void
gtk_list_box_insert (GtkListBox *box,
                     GtkWidget *child,
                     gint position);
```

Insert the `child` into the `box` at `position` . If a sort function is set, the widget will actually be inserted at the calculated position and this function has the same effect of `gtk_container_add()`.

If `position` is -1, or larger than the total number of items in the `box` , then the `child` will be appended to the end.

Parameters

box	a GtkListBox	
child	the GtkWidget to add	
position	the position to insert `child` in	

Since: 3.10

---

gtk_list_box_select_row ()

```
void
gtk_list_box_select_row (GtkListBox *box,
                         GtkListBoxRow *row);
```

Make `row` the currently selected row.

Parameters

box	a GtkListBox	
row	The row to select or `NULL`.	[allow-none]

Since: 3.10

---

gtk_list_box_unselect_row ()

```
void
gtk_list_box_unselect_row (GtkListBox *box,
                           GtkListBoxRow *row);
```

Unselects a single row of `box` , if the selection mode allows it.

Parameters

box	a GtkListBox	
row	the row to unselected	

Since: 3.14

---

gtk_list_box_select_all ()

```
void
gtk_list_box_select_all (GtkListBox *box);
```

Select all children of `box` , if the selection mode allows it.

Parameters

box	a GtkListBox	

Since: 3.14

---

gtk_list_box_unselect_all ()

```
void
gtk_list_box_unselect_all (GtkListBox *box);
```

Unselect all children of `box` , if the selection mode allows it.

Parameters

box	a GtkListBox	

Since: 3.14

---

gtk_list_box_get_selected_row ()

```
GtkListBoxRow *
gtk_list_box_get_selected_row (GtkListBox *box);
```

Gets the selected row.

Note that the box may allow multiple selection, in which case you should use `gtk_list_box_selected_foreach()` to find all selected rows.

Parameters

box	a GtkListBox	

Returns

the selected row.

[transfer none]

Since: 3.10

GtkListBoxForeachFunc ()

```
void
(*GtkListBoxForeachFunc) (GtkListBox *box,
                          GtkListBoxRow *row,
                          gpointer user_data);
```

A function used by `gtk_list_box_selected_foreach()`. It will be called on every selected child of the `box` .

Parameters

box	a GtkListBox	
row	a GtkListBoxRow	
user_data	user data.	[closure]

Since: 3.14

---

gtk_list_box_selected_foreach ()

```
void
gtk_list_box_selected_foreach (GtkListBox *box,
                               GtkListBoxForeachFunc func,
                               gpointer data);
```

Calls a function for each selected child.

Note that the selection cannot be modified from within this function.

Parameters

box	a GtkListBox
func	the function to call for each selected [scope call] child.
data	user data to pass to the function

Since: 3.14

---

gtk_list_box_get_selected_rows ()

```
GList *
gtk_list_box_get_selected_rows (GtkListBox *box);
```

Creates a list of all selected children.

Parameters

box	a GtkListBox

Returns

A GList containing the GtkWidget for each selected child. Free with `g_list_free()` when done.

[element-type GtkListBoxRow][transfer container]

Since: 3.14

---

gtk_list_box_set_selection_mode ()

```
void
gtk_list_box_set_selection_mode (GtkListBox *box,
                                 GtkSelectionMode mode);
```

Sets how selection works in the listbox. See GtkSelectionMode for details.

Parameters

box	a GtkListBox
mode	The GtkSelectionMode

Since: 3.10

---

gtk_list_box_get_selection_mode ()

```
GtkSelectionMode
gtk_list_box_get_selection_mode (GtkListBox *box);
```

Gets the selection mode of the listbox.

Parameters

box	a GtkListBox

Returns

a GtkSelectionMode

Since: 3.10

---

gtk_list_box_set_activate_on_single_click ()

```
void
gtk_list_box_set_activate_on_single_click
                                (GtkListBox *box,
                                 gboolean single);
```

If `single` is `TRUE`, rows will be activated when you click on them, otherwise you need to double-click.

Parameters

box	a GtkListBox
single	a boolean

Since: 3.10

---

gtk_list_box_get_activate_on_single_click ()

```
gboolean
gtk_list_box_get_activate_on_single_click
                                (GtkListBox *box);
```

Returns whether rows activate on single clicks.

Parameters

box	a GtkListBox	

Returns

`TRUE` if rows are activated on single click, `FALSE` otherwise

Since: 3.10

---

gtk_list_box_get_adjustment ()

```
GtkAdjustment *
gtk_list_box_get_adjustment (GtkListBox *box);
```

Gets the adjustment (if any) that the widget uses to for vertical scrolling.

Parameters

box	a GtkListBox	

Returns

the adjustment.

[transfer none]

Since: 3.10

---

gtk_list_box_set_adjustment ()

```
void
gtk_list_box_set_adjustment (GtkListBox *box,
                             GtkAdjustment *adjustment);
```

Sets the adjustment (if any) that the widget uses to for vertical scrolling. For instance, this is used to get the page size for PageUp/Down key handling.

In the normal case when the `box` is packed inside a GtkScrolledWindow the adjustment from that will be picked up automatically, so there is no need to manually do that.

Parameters

box	a GtkListBox	
adjustment	the adjustment, or `NULL`.	[allow-none]

Since: 3.10

---

gtk_list_box_set_placeholder ()

```
void
gtk_list_box_set_placeholder (GtkListBox *box,
                              GtkWidget *placeholder);
```

Sets the placeholder widget that is shown in the list when it doesn't display any visible children.

Parameters

box	a GtkListBox	
placeholder	a GtkWidget or `NULL`.	[allow-none]

Since: 3.10

---

gtk_list_box_get_row_at_index ()

```
GtkListBoxRow *
gtk_list_box_get_row_at_index (GtkListBox *box,
                               gint index_);
```

Gets the n-th child in the list (not counting headers). If `_index` is negative or larger than the number of items in the list, `NULL` is returned.

Parameters

box	a GtkListBox
index_	the index of the row

Returns

the child GtkWidget or `NULL`.

[transfer none][nullable]

Since: 3.10

---

gtk_list_box_get_row_at_y ()

```
GtkListBoxRow *
gtk_list_box_get_row_at_y (GtkListBox *box,
                           gint y);
```

Gets the row at the `y` position.

Parameters

box	a GtkListBox
y	position

Returns

the row or `NULL` in case no row exists for the given y coordinate.

[transfer none][nullable]

Since: 3.10

---

gtk_list_box_invalidate_filter ()

```
void
gtk_list_box_invalidate_filter (GtkListBox *box);
```

Update the filtering for all rows. Call this when result of the filter function on the `box` is changed due to an external factor. For instance, this would be used if the filter function just looked for a specific search string and the entry with the search string has changed.

Parameters

box	a GtkListBox

Since: 3.10

---

gtk_list_box_invalidate_headers ()

```
void
gtk_list_box_invalidate_headers (GtkListBox *box);
```

Update the separators for all rows. Call this when result of the header function on the `box` is changed due to an external factor.

Parameters

box	a GtkListBox

Since: 3.10

---

gtk_list_box_invalidate_sort ()

```
void
gtk_list_box_invalidate_sort (GtkListBox *box);
```

Update the sorting for all rows. Call this when result of the sort function on the `box` is changed due to an external factor.

Parameters

box	a GtkListBox

Since: 3.10

---

gtk_list_box_set_filter_func ()

```
void
gtk_list_box_set_filter_func (GtkListBox *box,
                              GtkListBoxFilterFunc filter_func,
                              gpointer user_data,
                              GDestroyNotify destroy);
```

By setting a filter function on the `box` one can decide dynamically which of the rows to show. For instance, to implement a search function on a list that filters the original list to only show the matching rows.

The `filter_func` will be called for each row after the call, and it will continue to be called each time a row changes (via `gtk_list_box_row_changed()`) or when `gtk_list_box_invalidate_filter()` is called.

Note that using a filter function is incompatible with using a model (see `gtk_list_box_bind_model()`).

Parameters

box	a GtkListBox	
filter_func	callback that lets you filter which rows to show.	[closure user_data][allow-none]
user_data	user data passed to `filter_func`	
destroy	destroy notifier for `user_data`	

Since: 3.10

---

gtk_list_box_set_header_func ()

```
void
gtk_list_box_set_header_func (GtkListBox *box,
                              GtkListBoxUpdateHeaderFunc update_header,
                              gpointer user_data,
                              GDestroyNotify destroy);
```

By setting a header function on the `box` one can dynamically add headers in front of rows, depending on the contents of the row and its position in the list. For instance, one could use it to add headers in front of the first item of a new kind, in a list sorted by the kind.

The `update_header` can look at the current header widget using `gtk_list_box_row_get_header()` and either update the state of the widget as needed, or set a new one using `gtk_list_box_row_set_header()`. If no header is needed, set the header to `NULL`.

Note that you may get many calls `update_header` to this for a particular row when e.g. changing things that don’t affect the header. In this case it is important for performance to not blindly replace an existing header with an identical one.

The `update_header` function will be called for each row after the call, and it will continue to be called each time a row changes (via `gtk_list_box_row_changed()`) and when the row before changes (either by `gtk_list_box_row_changed()` on the previous row, or when the previous row becomes a different row). It is also called for all rows when `gtk_list_box_invalidate_headers()` is called.

Parameters

box	a GtkListBox	
update_header	callback that lets you add row headers.	[closure user_data][allow-none]
user_data	user data passed to `update_header`	
destroy	destroy notifier for `user_data`	

Since: 3.10

---

gtk_list_box_set_sort_func ()

```
void
gtk_list_box_set_sort_func (GtkListBox *box,
                            GtkListBoxSortFunc sort_func,
                            gpointer user_data,
                            GDestroyNotify destroy);
```

By setting a sort function on the `box` one can dynamically reorder the rows of the list, based on the contents of the rows.

The `sort_func` will be called for each row after the call, and will continue to be called each time a row changes (via `gtk_list_box_row_changed()`) and when `gtk_list_box_invalidate_sort()` is called.

Note that using a sort function is incompatible with using a model (see `gtk_list_box_bind_model()`).

Parameters

box	a GtkListBox	
sort_func	the sort function.	[closure user_data][allow-none]
user_data	user data passed to `sort_func`	
destroy	destroy notifier for `user_data`	

Since: 3.10

---

gtk_list_box_drag_highlight_row ()

```
void
gtk_list_box_drag_highlight_row (GtkListBox *box,
                                 GtkListBoxRow *row);
```

This is a helper function for implementing DnD onto a GtkListBox. The passed in `row` will be highlighted via `gtk_drag_highlight()`, and any previously highlighted row will be unhighlighted.

The row will also be unhighlighted when the widget gets a drag leave event.

Parameters

box	a GtkListBox
row	a GtkListBoxRow

Since: 3.10

---

gtk_list_box_drag_unhighlight_row ()

```
void
gtk_list_box_drag_unhighlight_row (GtkListBox *box);
```

If a row has previously been highlighted via `gtk_list_box_drag_highlight_row()` it will have the highlight removed.

Parameters

box	a GtkListBox

Since: 3.10

GtkListBoxCreateWidgetFunc ()

```
GtkWidget *
(*GtkListBoxCreateWidgetFunc) (gpointer item,
                               gpointer user_data);
```

Called for list boxes that are bound to a GListModel with `gtk_list_box_bind_model()` for each item that gets added to the model.

Versions of GTK+ prior to 3.18 called `gtk_widget_show_all()` on the rows created by the GtkListBoxCreateWidgetFunc, but this forced all widgets inside the row to be shown, and is no longer the case. Applications should be updated to show the desired row widgets.

Parameters

item	the item from the model for which to create a widget for.	[type GObject]
user_data	user data.	[closure]

Returns

a GtkWidget that represents `item` .

[transfer full]

Since: 3.16

---

gtk_list_box_bind_model ()

```
void
gtk_list_box_bind_model (GtkListBox *box,
                         GListModel *model,
                         GtkListBoxCreateWidgetFunc create_widget_func,
                         gpointer user_data,
                         GDestroyNotify user_data_free_func);
```

Binds `model` to `box` .

If box was already bound to a model, that previous binding is destroyed.

The contents of box are cleared and then filled with widgets that represent items from model. box is updated whenever model changes. If model is NULL, box is left empty.

It is undefined to add or remove widgets directly (for example, with gtk_list_box_insert() or gtk_container_add()) while box is bound to a model.

Note that using a model is incompatible with the filtering and sorting functionality in GtkListBox. When using a model, filtering and sorting should be implemented by the model.

Parameters

box	a GtkListBox	
model	the GListModel to be bound to box.	[nullable]
create_widget_func	a function that creates widgets for items or NULL in case you also passed NULL as model.	[nullable]
user_data	user data passed to create_widget_func	
user_data_free_func	function for freeing user_data	

Since: 3.16

---

gtk_list_box_row_new ()

```
GtkWidget *
gtk_list_box_row_new (void);
```

Creates a new GtkListBoxRow, to be used as a child of a GtkListBox.

Returns

a new GtkListBoxRow

Since: 3.10

---

gtk_list_box_row_changed ()

```
void
gtk_list_box_row_changed (GtkListBoxRow *row);
```

Marks row as changed, causing any state that depends on this to be updated. This affects sorting, filtering and headers.

Note that calls to this method must be in sync with the data used for the row functions. For instance, if the list is mirroring some external data set, and *two* rows changed in the external data set then when you call gtk_list_box_row_changed() on the first row the sort function must only read the new data for the first of the two changed rows, otherwise the resorting of the rows will be wrong.

This generally means that if you don’t fully control the data model you have to duplicate the data that affects the listbox row functions into the row widgets themselves. Another alternative is to call

gtk_list_box_invalidate_sort() on any model change, but that is more expensive.

Parameters

row	a GtkListBoxRow

Since: 3.10

---

gtk_list_box_row_is_selected ()

```
gboolean
gtk_list_box_row_is_selected (GtkListBoxRow *row);
```

Returns whether the child is currently selected in its GtkListBox container.

Parameters

row	a GtkListBoxRow

Returns

TRUE if row is selected

Since: 3.14

---

gtk_list_box_row_get_header ()

```
GtkWidget *
gtk_list_box_row_get_header (GtkListBoxRow *row);
```

Returns the current header of the row . This can be used in a GtkListBoxUpdateHeaderFunc to see if there is a header set already, and if so to update the state of it.

Parameters

row	a GtkListBoxRow

Returns

the current header, or NULL if none.

[transfer none][nullable]

Since: 3.10

---

gtk_list_box_row_set_header ()

```
void
gtk_list_box_row_set_header (GtkListBoxRow *row,
                             GtkWidget *header);
```

Sets the current header of the `row` . This is only allowed to be called from a GtkListBoxUpdateHeaderFunc. It will replace any existing header in the row, and be shown in front of the row in the listbox.

Parameters

row	a GtkListBoxRow	
header	the header, or `NULL`.	[allow-none]

Since: 3.10

---

gtk_list_box_row_get_index ()

```
gint
gtk_list_box_row_get_index (GtkListBoxRow *row);
```

Gets the current index of the `row` in its GtkListBox container.

Parameters

row	a GtkListBoxRow

Returns

the index of the `row` , or -1 if the `row` is not in a listbox

Since: 3.10

---

gtk_list_box_row_set_activatable ()

```
void
gtk_list_box_row_set_activatable (GtkListBoxRow *row,
                                  gboolean activatable);
```

Set the “activatable” property for this row.

Parameters

row	a GtkListBoxRow
activatable	`TRUE` to mark the row as activatable

Since: 3.14

---

gtk_list_box_row_get_activatable ()

```
gboolean
gtk_list_box_row_get_activatable (GtkListBoxRow *row);
```

Gets the value of the “activatable” property for this row.

Parameters

row	a GtkListBoxRow

Returns

TRUE if the row is activatable

Since: 3.14

---

gtk_list_box_row_set_selectable ()

```
void
gtk_list_box_row_set_selectable (GtkListBoxRow *row,
                                 gboolean selectable);
```

Set the “selectable” property for this row.

Parameters

row	a GtkListBoxRow
selectable	TRUE to mark the row as selectable

Since: 3.14

---

gtk_list_box_row_get_selectable ()

```
gboolean
gtk_list_box_row_get_selectable (GtkListBoxRow *row);
```

Gets the value of the “selectable” property for this row.

Parameters

row	a GtkListBoxRow

Returns

TRUE if the row is selectable

Since: 3.14

Types and Values

struct GtkListBox

```
struct GtkListBox;
```

---

struct GtkListBoxClass

```
struct GtkListBoxClass {
  GtkContainerClass parent_class;


  void (*row_selected)        (GtkListBox      *box,
                               GtkListBoxRow   *row);
  void (*row_activated)       (GtkListBox      *box,
                               GtkListBoxRow   *row);
  void (*activate_cursor_row) (GtkListBox      *box);
  void (*toggle_cursor_row)   (GtkListBox      *box);
  void (*move_cursor)         (GtkListBox      *box,
                               GtkMovementStep  step,
                               gint             count);
  void (*selected_rows_changed) (GtkListBox    *box);
  void (*select_all)            (GtkListBox    *box);
  void (*unselect_all)          (GtkListBox    *box);
};
```

Members

`row_selected` ()	Class handler for the “row-selected” signal
`row_activated` ()	Class handler for the “row-activated” signal
`activate_cursor_row` ()	Class handler for the “activate-cursor-row” signal
`toggle_cursor_row` ()	Class handler for the “toggle-cursor-row” signal
`move_cursor` ()	Class handler for the “move-cursor” signal
`selected_rows_changed` ()	Class handler for the “selected-rows-changed” signal
`select_all` ()	Class handler for the “select-all” signal
`unselect_all` ()	Class handler for the “unselect-all” signal

---

struct GtkListBoxRow

```
struct GtkListBoxRow;
```

---

struct GtkListBoxRowClass

```
struct GtkListBoxRowClass {
  GtkBinClass parent_class;


  void (* activate) (GtkListBoxRow *row);
};
```

Members

`activate` ()

Property Details

The “activate-on-single-click” property

```
  “activate-on-single-click” gboolean
```

Activate row on a single click.

Flags: Read / Write

Default value: TRUE

The “selection-mode” property

```
  “selection-mode”           GtkSelectionMode
```

The selection mode.

Flags: Read / Write

Default value: GTK_SELECTION_SINGLE

The “activatable” property

```
  “activatable”              gboolean
```

The property determines whether the “row-activated” signal will be emitted for this row.

Flags: Read / Write

Default value: TRUE

Since: 3.14

The “selectable” property

```
  “selectable”               gboolean
```

The property determines whether this row can be selected.

Flags: Read / Write

Default value: TRUE

Since: 3.14

Signal Details

The “activate-cursor-row” signal

```
void
user_function (GtkListBox *listbox,
               gpointer    user_data)
```

Flags: Action

The “move-cursor” signal

```
void
user_function (GtkListBox     *listbox,
               GtkMovementStep arg1,
               gint            arg2,
               gpointer        user_data)
```

Flags: Action

The “row-activated” signal

```
void
user_function (GtkListBox    *box,
               GtkListBoxRow *row,
               gpointer       user_data)
```

The ::row-activated signal is emitted when a row has been activated by the user.

Parameters

box	the GtkListBox
row	the activated row
user_data	user data set when the signal handler was connected.

Flags: Run Last

Since: 3.10

The “row-selected” signal

```
void
user_function (GtkListBox    *box,
               GtkListBoxRow *row,
               gpointer       user_data)
```

The ::row-selected signal is emitted when a new row is selected, or (with a `NULL row`) when the selection is cleared.

When the `box` is using GTK_SELECTION_MULTIPLE, this signal will not give you the full picture of selection changes, and you should use the “selected-rows-changed” signal instead.

Parameters

box	the GtkListBox	
row	the selected row.	[nullable]
user_data	user data set when the signal handler was connected.	

Flags: Run Last

Since: 3.10

The “select-all” signal

```
void
user_function (GtkListBox *box,
               gpointer    user_data)
```

The ::select-all signal is a keybinding signal which gets emitted to select all children of the box, if the selection mode permits it.

The default bindings for this signal is Ctrl-a.

Parameters

box	the GtkListBox on which the signal is emitted
user_data	user data set when the signal handler was connected.

Flags: Action

Since: 3.14

The “selected-rows-changed” signal

```
void
user_function (GtkListBox *box,
               gpointer    user_data)
```

The ::selected-rows-changed signal is emitted when the set of selected rows changes.

Parameters

box	the GtkListBox on wich the signal is emitted
user_data	user data set when the signal handler was connected.

Flags: Run First

Since: 3.14

The “toggle-cursor-row” signal

```
void
user_function (GtkListBox *listbox,
               gpointer    user_data)
```

Flags: Action

The “unselect-all” signal

```
void
user_function (GtkListBox *box,
               gpointer    user_data)
```

The ::unselect-all signal is a keybinding signal which gets emitted to unselect all children of the box, if the selection mode permits it.

The default bindings for this signal is Ctrl-Shift-a.

Parameters

box	the GtkListBox on which the signal is emitted
user_data	user data set when the signal handler was connected.

Flags: Action

Since: 3.14

The “activate” signal

```
void
user_function (GtkListBoxRow *listboxrow,
               gpointer       user_data)
```

This is a keybinding signal, which will cause this row to be activated.

If you want to be notified when the user activates a row (by key or not), use the “row-activated” signal on the row’s parent GtkListBox.

Parameters

user_data	user data set when the signal handler was connected.

Flags: Action

Since: 3.10

See Also

GtkScrolledWindow

GtkFlowBox

GtkFlowBox — A container that allows reflowing its children

Functions

GtkWidget *	gtk_flow_box_new ()
void	gtk_flow_box_insert ()
GtkFlowBoxChild *	gtk_flow_box_get_child_at_index ()
GtkFlowBoxChild *	gtk_flow_box_get_child_at_pos ()
void	gtk_flow_box_set_hadjustment ()
void	gtk_flow_box_set_vadjustment ()
void	gtk_flow_box_set_homogeneous ()
gboolean	gtk_flow_box_get_homogeneous ()
void	gtk_flow_box_set_row_spacing ()
guint	gtk_flow_box_get_row_spacing ()
void	gtk_flow_box_set_column_spacing ()
guint	gtk_flow_box_get_column_spacing ()
void	gtk_flow_box_set_min_children_per_line ()
guint	gtk_flow_box_get_min_children_per_line ()
void	gtk_flow_box_set_max_children_per_line ()
guint	gtk_flow_box_get_max_children_per_line ()
void	gtk_flow_box_set_activate_on_single_click ()
gboolean	gtk_flow_box_get_activate_on_single_click ()
void	(*GtkFlowBoxForeachFunc) ()
void	gtk_flow_box_selected_foreach ()
GList *	gtk_flow_box_get_selected_children ()

void	gtk_flow_box_select_child ()
void	gtk_flow_box_unselect_child ()
void	gtk_flow_box_select_all ()
void	gtk_flow_box_unselect_all ()
void	gtk_flow_box_set_selection_mode ()
GtkSelectionMode	gtk_flow_box_get_selection_mode ()
gboolean	(*GtkFlowBoxFilterFunc) ()
void	gtk_flow_box_set_filter_func ()
void	gtk_flow_box_invalidate_filter ()
gint	(*GtkFlowBoxSortFunc) ()
void	gtk_flow_box_set_sort_func ()
void	gtk_flow_box_invalidate_sort ()
GtkWidget *	(*GtkFlowBoxCreateWidgetFunc) ()
void	gtk_flow_box_bind_model ()
GtkWidget *	gtk_flow_box_child_new ()
gint	gtk_flow_box_child_get_index ()
gboolean	gtk_flow_box_child_is_selected ()
void	gtk_flow_box_child_changed ()

Properties

gboolean	activate-on-single-click	Read / Write
guint	column-spacing	Read / Write
gboolean	homogeneous	Read / Write
guint	max-children-per-line	Read / Write
guint	min-children-per-line	Read / Write
guint	row-spacing	Read / Write
GtkSelectionMode	selection-mode	Read / Write

Signals

void	activate-cursor-child	Action
void	child-activated	Run Last
gboolean	move-cursor	Action
void	select-all	Action
void	selected-children-changed	Run First
void	toggle-cursor-child	Action
void	unselect-all	Action
void	activate	Action

Types and Values

struct	GtkFlowBox
struct	GtkFlowBoxChild

Object Hierarchy

```
GObject
└── GInitiallyUnowned
```

```
        └── GtkWidget
            └── GtkContainer
                ├── GtkBin
                │   └── GtkFlowBoxChild
                └── GtkFlowBox
```

Implemented Interfaces

GtkFlowBox implements AtkImplementorIface, GtkBuildable and GtkOrientable.

GtkFlowBoxChild implements AtkImplementorIface and GtkBuildable.

Includes

```
#include <gtk/gtk.h>
```

Description

A GtkFlowBox positions child widgets in sequence according to its orientation.

For instance, with the horizontal orientation, the widgets will be arranged from left to right, starting a new row under the previous row when necessary. Reducing the width in this case will require more rows, so a larger height will be requested.

Likewise, with the vertical orientation, the widgets will be arranged from top to bottom, starting a new column to the right when necessary. Reducing the height will require more columns, so a larger width will be requested.

The size request of a GtkFlowBox alone may not be what you expect; if you need to be able to shrink it along both axes and dynamically reflow its children, you may have to wrap it in a GtkScrolledWindow to enable that.

The children of a GtkFlowBox can be dynamically sorted and filtered.

Although a GtkFlowBox must have only GtkFlowBoxChild children, you can add any kind of widget to it via `gtk_container_add()`, and a GtkFlowBoxChild widget will automatically be inserted between the box and the widget.

Also see GtkListBox.

GtkFlowBox was added in GTK+ 3.12.

CSS nodes

```
1  flowbox
2  ├── flowboxchild
3  │   └── <child>
4  ├── flowboxchild
5  │   └── <child>
6  ┊
7  └── [rubberband]
```

GtkFlowBox uses a single CSS node with name flowbox. GtkFlowBoxChild uses a single CSS node with name flowboxchild. For rubberband selection, a subnode with name rubberband is used.

Functions

gtk_flow_box_new ()

```
GtkWidget *
gtk_flow_box_new (void);
```

Creates a GtkFlowBox.

Returns

a new GtkFlowBox container

Since: 3.12

---

gtk_flow_box_insert ()

```
void
gtk_flow_box_insert (GtkFlowBox *box,
                     GtkWidget *widget,
                     gint position);
```

Inserts the `widget` into `box` at `position` .

If a sort function is set, the widget will actually be inserted at the calculated position and this function has the same effect as `gtk_container_add()`.

If `position` is -1, or larger than the total number of children in the `box` , then the `widget` will be appended to the end.

Parameters

box	a GtkFlowBox
widget	the GtkWidget to add
position	the position to insert `child` in

Since: 3.12

---

gtk_flow_box_get_child_at_index ()

```
GtkFlowBoxChild *
gtk_flow_box_get_child_at_index (GtkFlowBox *box,
                                 gint idx);
```

Gets the nth child in the `box` .

Parameters

box	a GtkFlowBox
idx	the position of the child

Returns

the child widget, which will always be a GtkFlowBoxChild or `NULL` in case no child widget with the given index exists.

[transfer none][nullable]

Since: 3.12

---

gtk_flow_box_get_child_at_pos ()

```
GtkFlowBoxChild *
gtk_flow_box_get_child_at_pos (GtkFlowBox *box,
                               gint x,
                               gint y);
```

Gets the child in the (`x` , `y`) position.

Parameters

box	a GtkFlowBox
x	the x coordinate of the child
y	the y coordinate of the child

Returns

the child widget, which will always be a GtkFlowBoxChild or `NULL` in case no child widget exists for the given x and y coordinates.

[transfer none][nullable]

Since: 3.22.6

---

gtk_flow_box_set_hadjustment ()

```
void
gtk_flow_box_set_hadjustment (GtkFlowBox *box,
                              GtkAdjustment *adjustment);
```

Hooks up an adjustment to focus handling in `box` . The adjustment is also used for autoscrolling during rubberband selection. See `gtk_scrolled_window_get_hadjustment()` for a typical way of obtaining the adjustment, and `gtk_flow_box_set_vadjustment()`for setting the vertical adjustment.

The adjustments have to be in pixel units and in the same coordinate system as the allocation for immediate children of the box.

Parameters

box	a GtkFlowBox

adjustment	an adjustment which should be adjusted when the focus is moved among the descendents of `container`

Since: 3.12

---

gtk_flow_box_set_vadjustment ()

```
void
gtk_flow_box_set_vadjustment (GtkFlowBox *box,
                              GtkAdjustment *adjustment);
```

Hooks up an adjustment to focus handling in `box` . The adjustment is also used for autoscrolling during rubberband selection. See `gtk_scrolled_window_get_vadjustment()` for a typical way of obtaining the adjustment, and `gtk_flow_box_set_hadjustment()` for setting the horizontal adjustment.

The adjustments have to be in pixel units and in the same coordinate system as the allocation for immediate children of the box.

Parameters

box	a GtkFlowBox
adjustment	an adjustment which should be adjusted when the focus is moved among the descendents of `container`

Since: 3.12

---

gtk_flow_box_set_homogeneous ()

```
void
gtk_flow_box_set_homogeneous (GtkFlowBox *box,
                              gboolean homogeneous);
```

Sets the “homogeneous” property of `box` , controlling whether or not all children of `box` are given equal space in the box.

Parameters

box	a GtkFlowBox
homogeneous	`TRUE` to create equal allotments, `FALSE` for variable allotments

Since: 3.12

---

gtk_flow_box_get_homogeneous ()

```
gboolean
gtk_flow_box_get_homogeneous (GtkFlowBox *box);
```

Returns whether the box is homogeneous (all children are the same size). See `gtk_box_set_homogeneous()`.

Parameters

box	a GtkFlowBox

Returns

`TRUE` if the box is homogeneous.

Since: 3.12

---

gtk_flow_box_set_row_spacing ()

```
void
gtk_flow_box_set_row_spacing (GtkFlowBox *box,
                              guint spacing);
```

Sets the vertical space to add between children. See the "row-spacing" property.

Parameters

box	a GtkFlowBox
spacing	the spacing to use

Since: 3.12

---

gtk_flow_box_get_row_spacing ()

```
guint
gtk_flow_box_get_row_spacing (GtkFlowBox *box);
```

Gets the vertical spacing.

Parameters

box	a GtkFlowBox

Returns

the vertical spacing

Since: 3.12

---

gtk_flow_box_set_column_spacing ()

```
void
```

```
gtk_flow_box_set_column_spacing (GtkFlowBox *box,
                                 guint spacing);
```
Sets the horizontal space to add between children. See the “column-spacing” property.

Parameters

box	a GtkFlowBox
spacing	the spacing to use

Since: 3.12

---

gtk_flow_box_get_column_spacing ()

```
guint
gtk_flow_box_get_column_spacing (GtkFlowBox *box);
```
Gets the horizontal spacing.

Parameters

box	a GtkFlowBox

Returns

the horizontal spacing

Since: 3.12

---

gtk_flow_box_set_min_children_per_line ()

```
void
gtk_flow_box_set_min_children_per_line
                              (GtkFlowBox *box,
                               guint n_children);
```
Sets the minimum number of children to line up in `box` ’s orientation before flowing.

Parameters

box	a GtkFlowBox
n_children	the minimum number of children per line

Since: 3.12

---

gtk_flow_box_get_min_children_per_line ()

```
guint
gtk_flow_box_get_min_children_per_line
```

```
                                (GtkFlowBox *box);
```

Gets the minimum number of children per line.

Parameters

box	a GtkFlowBox

Returns

the minimum number of children per line

Since: 3.12

---

gtk_flow_box_set_max_children_per_line ()

```
void
gtk_flow_box_set_max_children_per_line
                                (GtkFlowBox *box,
                                 guint n_children);
```

Sets the maximum number of children to request and allocate space for in `box` 's orientation.

Setting the maximum number of children per line limits the overall natural size request to be no more than `n_children` children long in the given orientation.

Parameters

box	a GtkFlowBox
n_children	the maximum number of children per line

Since: 3.12

---

gtk_flow_box_get_max_children_per_line ()

```
guint
gtk_flow_box_get_max_children_per_line
                                (GtkFlowBox *box);
```

Gets the maximum number of children per line.

Parameters

box	a GtkFlowBox

Returns

the maximum number of children per line

Since: 3.12

gtk_flow_box_set_activate_on_single_click ()

```
void
gtk_flow_box_set_activate_on_single_click
                                (GtkFlowBox *box,
                                 gboolean single);
```

If `single` is `TRUE`, children will be activated when you click on them, otherwise you need to double-click.

Parameters

box	a GtkFlowBox
single	`TRUE` to emit child-activated on a single click

Since: 3.12

---

gtk_flow_box_get_activate_on_single_click ()

```
gboolean
gtk_flow_box_get_activate_on_single_click
                                (GtkFlowBox *box);
```

Returns whether children activate on single clicks.

Parameters

box	a GtkFlowBox

Returns

`TRUE` if children are activated on single click, `FALSE` otherwise

Since: 3.12

GtkFlowBoxForeachFunc ()

```
void
(*GtkFlowBoxForeachFunc) (GtkFlowBox *box,
                          GtkFlowBoxChild *child,
                          gpointer user_data);
```

A function used by `gtk_flow_box_selected_foreach()`. It will be called on every selected child of the `box` .

Parameters

box	a GtkFlowBox
child	a GtkFlowBoxChild

user_data	user data.	[closure]

Since: 3.12

---

gtk_flow_box_selected_foreach ()

```
void
gtk_flow_box_selected_foreach (GtkFlowBox *box,
                               GtkFlowBoxForeachFunc func,
                               gpointer data);
```

Calls a function for each selected child.

Note that the selection cannot be modified from within this function.

Parameters

box	a GtkFlowBox	
func	the function to call for each selected child.	[scope call]
data	user data to pass to the function	

Since: 3.12

---

gtk_flow_box_get_selected_children ()

```
GList *
gtk_flow_box_get_selected_children (GtkFlowBox *box);
```

Creates a list of all selected children.

Parameters

box	a GtkFlowBox

Returns

A GList containing the GtkWidget for each selected child. Free with `g_list_free()` when done.

[element-type GtkFlowBoxChild][transfer container]

Since: 3.12

---

gtk_flow_box_select_child ()

```
void
gtk_flow_box_select_child (GtkFlowBox *box,
                           GtkFlowBoxChild *child);
```

Selects a single child of `box` , if the selection mode allows it.

Parameters

box	a GtkFlowBox
child	a child of `box`

Since: 3.12

---

gtk_flow_box_unselect_child ()

```
void
gtk_flow_box_unselect_child (GtkFlowBox *box,
                             GtkFlowBoxChild *child);
```

Unselects a single child of `box` , if the selection mode allows it.

Parameters

box	a GtkFlowBox
child	a child of `box`

Since: 3.12

---

gtk_flow_box_select_all ()

```
void
gtk_flow_box_select_all (GtkFlowBox *box);
```

Select all children of `box` , if the selection mode allows it.

Parameters

box	a GtkFlowBox

Since: 3.12

---

gtk_flow_box_unselect_all ()

```
void
gtk_flow_box_unselect_all (GtkFlowBox *box);
```

Unselect all children of `box` , if the selection mode allows it.

Parameters

box	a GtkFlowBox

Since: 3.12

---

gtk_flow_box_set_selection_mode ()

```
void
gtk_flow_box_set_selection_mode (GtkFlowBox *box,
                                 GtkSelectionMode mode);
```

Sets how selection works in `box` . See GtkSelectionMode for details.

Parameters

box	a GtkFlowBox
mode	the new selection mode

Since: 3.12

---

gtk_flow_box_get_selection_mode ()

```
GtkSelectionMode
gtk_flow_box_get_selection_mode (GtkFlowBox *box);
```

Gets the selection mode of `box` .

Parameters

box	a GtkFlowBox

Returns

the GtkSelectionMode

Since: 3.12

GtkFlowBoxFilterFunc ()

```
gboolean
(*GtkFlowBoxFilterFunc) (GtkFlowBoxChild *child,
                         gpointer user_data);
```

A function that will be called whenever a child changes or is added. It lets you control if the child should be visible or not.

Parameters

child	a GtkFlowBoxChild that may be filtered	
user_data	user data.	[closure]

Returns

`TRUE` if the row should be visible, `FALSE` otherwise

Since: 3.12

---

gtk_flow_box_set_filter_func ()

```
void
gtk_flow_box_set_filter_func (GtkFlowBox *box,
                              GtkFlowBoxFilterFunc filter_func,
                              gpointer user_data,
                              GDestroyNotify destroy);
```

By setting a filter function on the `box` one can decide dynamically which of the children to show. For instance, to implement a search function that only shows the children matching the search terms.

The `filter_func` will be called for each child after the call, and it will continue to be called each time a child changes (via `gtk_flow_box_child_changed()`) or when `gtk_flow_box_invalidate_filter()` is called.

Note that using a filter function is incompatible with using a model (see `gtk_flow_box_bind_model()`).

Parameters

box	a GtkFlowBox	
filter_func	callback that lets you filter which children to show.	[closure user_data][allow-none]
user_data	user data passed to `filter_func`	
destroy	destroy notifier for `user_data`	

Since: 3.12

---

gtk_flow_box_invalidate_filter ()

```
void
gtk_flow_box_invalidate_filter (GtkFlowBox *box);
```

Updates the filtering for all children.

Call this function when the result of the filter function on the `box` is changed due ot an external factor. For instance, this would be used if the filter function just looked for a specific search term, and the entry with the string has changed.

Parameters

box	a GtkFlowBox

Since: 3.12

GtkFlowBoxSortFunc ()

```
gint
(*GtkFlowBoxSortFunc) (GtkFlowBoxChild *child1,
                       GtkFlowBoxChild *child2,
```

```
                        gpointer user_data);
```
A function to compare two children to determine which should come first.

Parameters

child1	the first child	
child2	the second child	
user_data	user data.	[closure]

Returns

< 0 if `child1` should be before `child2` , 0 if the are equal, and > 0 otherwise

Since: 3.12

---

gtk_flow_box_set_sort_func ()

```
void
gtk_flow_box_set_sort_func (GtkFlowBox *box,
                            GtkFlowBoxSortFunc sort_func,
                            gpointer user_data,
                            GDestroyNotify destroy);
```
By setting a sort function on the `box` , one can dynamically reorder the children of the box, based on the contents of the children.

The `sort_func` will be called for each child after the call, and will continue to be called each time a child changes (via `gtk_flow_box_child_changed()`) and when `gtk_flow_box_invalidate_sort()` is called.

Note that using a sort function is incompatible with using a model (see `gtk_flow_box_bind_model()`).

Parameters

box	a GtkFlowBox	
sort_func	the sort function.	[closure user_data][allow-none]
user_data	user data passed to `sort_func`	
destroy	destroy notifier for `user_data`	

Since: 3.12

---

gtk_flow_box_invalidate_sort ()

```
void
gtk_flow_box_invalidate_sort (GtkFlowBox *box);
```
Updates the sorting for all children.

Call this when the result of the sort function on `box` is changed due to an external factor.

Parameters

box	a GtkFlowBox

Since: 3.12

GtkFlowBoxCreateWidgetFunc ()

```
GtkWidget *
(*GtkFlowBoxCreateWidgetFunc) (gpointer item,
                               gpointer user_data);
```

Called for flow boxes that are bound to a GListModel with `gtk_flow_box_bind_model()` for each item that gets added to the model.

Parameters

item	the item from the model for which to create a widget for.	[type GObject]
user_data	user data from `gtk_flow_box_bind_model()`.	[closure]

Returns

a GtkWidget that represents `item` .

[transfer full]

Since: 3.18

---

gtk_flow_box_bind_model ()

```
void
gtk_flow_box_bind_model (GtkFlowBox *box,
                         GListModel *model,
                         GtkFlowBoxCreateWidgetFunc create_widget_func,
                         gpointer user_data,
                         GDestroyNotify user_data_free_func);
```

Binds `model` to `box` .

If `box` was already bound to a model, that previous binding is destroyed.

The contents of `box` are cleared and then filled with widgets that represent items from `model` . `box` is updated whenever `model` changes. If `model` is `NULL`, `box` is left empty.

It is undefined to add or remove widgets directly (for example, with `gtk_flow_box_insert()` or `gtk_container_add()`) while `box` is bound to a model.

Note that using a model is incompatible with the filtering and sorting functionality in GtkFlowBox. When using a model, filtering and sorting should be implemented by the model.

Parameters

box	a GtkFlowBox
model	the GListModel to be bound to `box` . [allow-none]
create_widget_func	a function that creates widgets for items
user_data	user data passed to `create_widget_func`
user_data_free_func	function for freeing `user_data`

Since: 3.18

---

gtk_flow_box_child_new ()

```
GtkWidget *
gtk_flow_box_child_new (void);
```

Creates a new GtkFlowBoxChild, to be used as a child of a GtkFlowBox.

Returns

a new GtkFlowBoxChild

Since: 3.12

---

gtk_flow_box_child_get_index ()

```
gint
gtk_flow_box_child_get_index (GtkFlowBoxChild *child);
```

Gets the current index of the `child` in its GtkFlowBox container.

Parameters

child	a GtkFlowBoxChild

Returns

the index of the `child` , or -1 if the `child` is not in a flow box.

Since: 3.12

---

gtk_flow_box_child_is_selected ()

```
gboolean
gtk_flow_box_child_is_selected (GtkFlowBoxChild *child);
```

Returns whether the `child` is currently selected in its GtkFlowBox container.

Parameters

child	a GtkFlowBoxChild

Returns

`TRUE` if `child` is selected

Since: 3.12

---

gtk_flow_box_child_changed ()

```
void
gtk_flow_box_child_changed (GtkFlowBoxChild *child);
```

Marks `child` as changed, causing any state that depends on this to be updated. This affects sorting and filtering.

Note that calls to this method must be in sync with the data used for the sorting and filtering functions. For instance, if the list is mirroring some external data set, and *two* children changed in the external data set when you call `gtk_flow_box_child_changed()` on the first child, the sort function must only read the new data for the first of the two changed children, otherwise the resorting of the children will be wrong.

This generally means that if you don't fully control the data model, you have to duplicate the data that affects the sorting and filtering functions into the widgets themselves. Another alternative is to call `gtk_flow_box_invalidate_sort()` on any model change, but that is more expensive.

Parameters

child	a GtkFlowBoxChild

Since: 3.12

Types and Values

struct GtkFlowBox

```
struct GtkFlowBox;
```

---

struct GtkFlowBoxChild

```
struct GtkFlowBoxChild;
```

Property Details

The “activate-on-single-click” property

“activate-on-single-click” gboolean

Determines whether children can be activated with a single click, or require a double-click.

Flags: Read / Write

Default value: TRUE

The “column-spacing” property

“column-spacing” guint

The amount of horizontal space between two children.

Flags: Read / Write

Default value: 0

The “homogeneous” property

“homogeneous” gboolean

Determines whether all children should be allocated the same size.

Flags: Read / Write

Default value: FALSE

The “max-children-per-line” property

“max-children-per-line” guint

The maximum amount of children to request space for consecutively in the given orientation.

Flags: Read / Write

Allowed values: >= 1

Default value: 7

The “min-children-per-line” property

“min-children-per-line” guint

The minimum number of children to allocate consecutively in the given orientation.

Setting the minimum children per line ensures that a reasonably small height will be requested for the overall minimum width of the box.

Flags: Read / Write

Default value: 0

The “row-spacing” property

```
  “row-spacing”              guint
```

The amount of vertical space between two children.

Flags: Read / Write

Default value: 0

The “selection-mode” property

```
  “selection-mode”           GtkSelectionMode
```

The selection mode used by the flow box.

Flags: Read / Write

Default value: GTK_SELECTION_SINGLE

Signal Details

The “activate-cursor-child” signal

```
void
user_function (GtkFlowBox *box,
               gpointer    user_data)
```

The ::activate-cursor-child signal is a keybinding signal which gets emitted when the user activates the `box` .

Parameters

box	the GtkFlowBox on which the signal is emitted
user_data	user data set when the signal handler was connected.

Flags: Action

The “child-activated” signal

```
void
user_function (GtkFlowBox      *box,
               GtkFlowBoxChild *child,
               gpointer         user_data)
```

The ::child-activated signal is emitted when a child has been activated by the user.

Parameters

box	the GtkFlowBox on which the signal is emitted
child	the child that is activated
user_data	user data set when the signal handler was connected.

Flags: Run Last

The `"move-cursor"` signal

```
gboolean
user_function (GtkFlowBox     *box,
               GtkMovementStep step,
               gint            count,
               gpointer        user_data)
```

The ::move-cursor signal is a keybinding signal which gets emitted when the user initiates a cursor movement.

Applications should not connect to it, but may emit it with `g_signal_emit_by_name()` if they need to control the cursor programmatically.

The default bindings for this signal come in two variants, the variant with the Shift modifier extends the selection, the variant without the Shift modifer does not. There are too many key combinations to list them all here.

- Arrow keys move by individual children
- Home/End keys move to the ends of the box
- PageUp/PageDown keys move vertically by pages

Parameters

box	the GtkFlowBox on which the signal is emitted
step	the granularity fo the move, as a GtkMovementStep
count	the number of `step` units to move
user_data	user data set when the signal handler was connected.

Returns

`TRUE` to stop other handlers from being invoked for the event. `FALSE` to propagate the event further.

Flags: Action

The `"select-all"` signal

```
void
user_function (GtkFlowBox *box,
```

```
                gpointer    user_data)
```

The ::select-all signal is a keybinding signal which gets emitted to select all children of the box, if the selection mode permits it.

The default bindings for this signal is Ctrl-a.

Parameters

box	the GtkFlowBox on which the signal is emitted
user_data	user data set when the signal handler was connected.

Flags: Action

The `"selected-children-changed"` signal

```
void
user_function (GtkFlowBox *box,
               gpointer    user_data)
```

The ::selected-children-changed signal is emitted when the set of selected children changes.

Use `gtk_flow_box_selected_foreach()` or `gtk_flow_box_get_selected_children()` to obtain the selected children.

Parameters

box	the GtkFlowBox on wich the signal is emitted
user_data	user data set when the signal handler was connected.

Flags: Run First

The `"toggle-cursor-child"` signal

```
void
user_function (GtkFlowBox *box,
               gpointer    user_data)
```

The ::toggle-cursor-child signal is a keybinding signal which toggles the selection of the child that has the focus.

The default binding for this signal is Ctrl-Space.

Parameters

box	the GtkFlowBox on which the signal is emitted
user_data	user data set when the signal

	handler was connected.

Flags: Action

The “unselect-all” signal

```
void
user_function (GtkFlowBox *box,
               gpointer    user_data)
```

The ::unselect-all signal is a keybinding signal which gets emitted to unselect all children of the box, if the selection mode permits it.

The default bindings for this signal is Ctrl-Shift-a.

Parameters

box	the GtkFlowBox on which the signal is emitted
user_data	user data set when the signal handler was connected.

Flags: Action

The “activate” signal

```
void
user_function (GtkFlowBoxChild *child,
               gpointer         user_data)
```

The ::activate signal is emitted when the user activates a child widget in a GtkFlowBox, either by clicking or double-clicking, or by using the Space or Enter key.

While this signal is used as a keybinding signal, it can be used by applications for their own purposes.

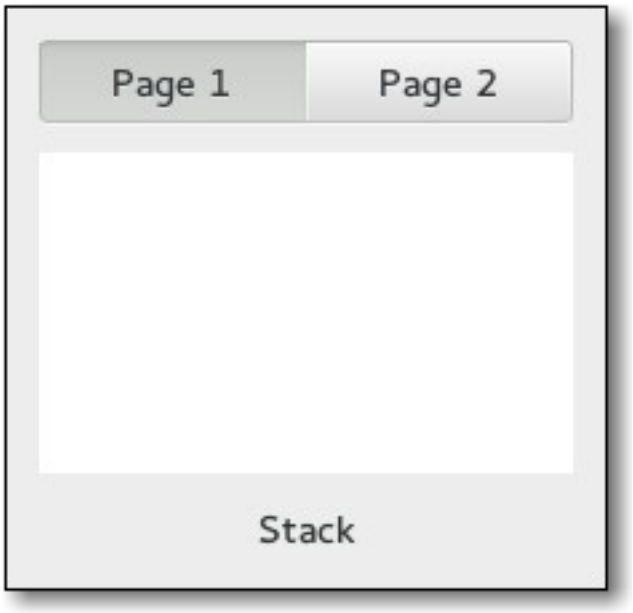
Parameters

child	The child on which the signal is emitted
user_data	user data set when the signal handler was connected.

Flags: Action

GtkStack

GtkStack — A stacking container



Functions

GtkWidget *	gtk_stack_new ()
void	gtk_stack_add_named ()
void	gtk_stack_add_titled ()
GtkWidget *	gtk_stack_get_child_by_name ()
void	gtk_stack_set_visible_child ()
GtkWidget *	gtk_stack_get_visible_child ()
void	gtk_stack_set_visible_child_name ()
const gchar *	gtk_stack_get_visible_child_name ()
void	gtk_stack_set_visible_child_full ()
void	gtk_stack_set_homogeneous ()
gboolean	gtk_stack_get_homogeneous ()
void	gtk_stack_set_hhomogeneous ()
gboolean	gtk_stack_get_hhomogeneous ()
void	gtk_stack_set_vhomogeneous ()
gboolean	gtk_stack_get_vhomogeneous ()
void	gtk_stack_set_transition_duration ()
guint	gtk_stack_get_transition_duration ()
void	gtk_stack_set_transition_type ()
GtkStackTransitionType	gtk_stack_get_transition_type ()
gboolean	gtk_stack_get_transition_running ()
gboolean	gtk_stack_get_interpolate_size ()
void	gtk_stack_set_interpolate_size ()

Properties

gboolean	hhomogeneous	Read / Write
gboolean	homogeneous	Read / Write
gboolean	interpolate-size	Read / Write
guint	transition-duration	Read / Write
gboolean	transition-running	Read
GtkStackTransitionType	transition-type	Read / Write

gboolean	vhomogeneous	Read / Write
GtkWidget *	visible-child	Read / Write
gchar *	visible-child-name	Read / Write

Child Properties

gchar *	icon-name	Read / Write
gchar *	name	Read / Write
gboolean	needs-attention	Read / Write
gint	position	Read / Write
gchar *	title	Read / Write

Types and Values

struct	GtkStack
enum	GtkStackTransitionType

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkContainer
            └── GtkStack
```

Implemented Interfaces

GtkStack implements AtkImplementorIface and GtkBuildable.

Includes

```
#include <gtk/gtk.h>
```

Description

The GtkStack widget is a container which only shows one of its children at a time. In contrast to GtkNotebook, GtkStack does not provide a means for users to change the visible child. Instead, the GtkStackSwitcher widget can be used with GtkStack to provide this functionality.

Transitions between pages can be animated as slides or fades. This can be controlled with `gtk_stack_set_transition_type()`. These animations respect the "gtk-enable-animations" setting.

The GtkStack widget was added in GTK+ 3.10.

CSS nodes

GtkStack has a single CSS node named stack.

Functions

gtk_stack_new ()

```
GtkWidget *
gtk_stack_new (void);
```

Creates a new GtkStack container.

Returns

a new GtkStack

Since: 3.10

---

gtk_stack_add_named ()

```
void
gtk_stack_add_named (GtkStack *stack,
                     GtkWidget *child,
                     const gchar *name);
```

Adds a child to `stack` . The child is identified by the `name` .

Parameters

stack	a GtkStack
child	the widget to add
name	the name for `child`

Since: 3.10

---

gtk_stack_add_titled ()

```
void
gtk_stack_add_titled (GtkStack *stack,
                      GtkWidget *child,
                      const gchar *name,
                      const gchar *title);
```

Adds a child to `stack` . The child is identified by the `name` . The `title` will be used by GtkStackSwitcher to represent `child` in a tab bar, so it should be short.

Parameters

stack	a GtkStack
child	the widget to add
name	the name for `child`
title	a human-readable title for `child`

Since: 3.10

gtk_stack_get_child_by_name ()

```
GtkWidget *
gtk_stack_get_child_by_name (GtkStack *stack,
                             const gchar *name);
```

Finds the child of the GtkStack with the name given as the argument. Returns `NULL` if there is no child with this name.

Parameters

stack	a GtkStack
name	the name of the child to find

Returns

the requested child of the GtkStack.

[transfer none][nullable]

Since: 3.12

---

gtk_stack_set_visible_child ()

```
void
gtk_stack_set_visible_child (GtkStack *stack,
                             GtkWidget *child);
```

Makes `child` the visible child of `stack` .

If `child` is different from the currently visible child, the transition between the two will be animated with the current transition type of `stack` .

Note that the `child` widget has to be visible itself (see `gtk_widget_show()`) in order to become the visible child of `stack` .

Parameters

stack	a GtkStack
child	a child of `stack`

Since: 3.10

---

gtk_stack_get_visible_child ()

```
GtkWidget *
gtk_stack_get_visible_child (GtkStack *stack);
```

Gets the currently visible child of `stack` , or `NULL` if there are no visible children.

Parameters

stack	a GtkStack

Returns

the visible child of the GtkStack.

[transfer none][nullable]

Since: 3.10

---

gtk_stack_set_visible_child_name ()

```
void
gtk_stack_set_visible_child_name (GtkStack *stack,
                                  const gchar *name);
```

Makes the child with the given name visible.

If `child` is different from the currently visible child, the transition between the two will be animated with the current transition type of `stack` .

Note that the child widget has to be visible itself (see `gtk_widget_show()`) in order to become the visible child of `stack` .

Parameters

stack	a GtkStack
name	the name of the child to make visible

Since: 3.10

---

gtk_stack_get_visible_child_name ()

```
const gchar *
gtk_stack_get_visible_child_name (GtkStack *stack);
```

Returns the name of the currently visible child of `stack` , or `NULL` if there is no visible child.

Parameters

stack	a GtkStack

Returns

the name of the visible child of the GtkStack.

[transfer none][nullable]

Since: 3.10

---

gtk_stack_set_visible_child_full ()

```
void
gtk_stack_set_visible_child_full (GtkStack *stack,
                                  const gchar *name,
                                  GtkStackTransitionType transition);
```

Makes the child with the given name visible.

Note that the child widget has to be visible itself (see `gtk_widget_show()`) in order to become the visible child of `stack`.

Parameters

stack	a GtkStack
name	the name of the child to make visible
transition	the transition type to use

Since: 3.10

---

gtk_stack_set_homogeneous ()

```
void
gtk_stack_set_homogeneous (GtkStack *stack,
                           gboolean homogeneous);
```

Sets the GtkStack to be homogeneous or not. If it is homogeneous, the GtkStack will request the same size for all its children. If it isn't, the stack may change size when a different child becomes visible.

Since 3.16, homogeneity can be controlled separately for horizontal and vertical size, with the “hhomogeneous” and “vhomogeneous”.

Parameters

stack	a GtkStack
homogeneous	`TRUE` to make `stack` homogeneous

Since: 3.10

---

gtk_stack_get_homogeneous ()

```
gboolean
gtk_stack_get_homogeneous (GtkStack *stack);
```

Gets whether `stack` is homogeneous. See `gtk_stack_set_homogeneous()`.

Parameters

stack	a GtkStack

Returns

whether `stack` is homogeneous.

Since: 3.10

---

gtk_stack_set_hhomogeneous ()

```
void
gtk_stack_set_hhomogeneous (GtkStack *stack,
                            gboolean hhomogeneous);
```

Sets the GtkStack to be horizontally homogeneous or not. If it is homogeneous, the GtkStack will request the same width for all its children. If it isn't, the stack may change width when a different child becomes visible.

Parameters

stack	a GtkStack
hhomogeneous	`TRUE` to make `stack` horizontally homogeneous

Since: 3.16

---

gtk_stack_get_hhomogeneous ()

```
gboolean
gtk_stack_get_hhomogeneous (GtkStack *stack);
```

Gets whether `stack` is horizontally homogeneous. See `gtk_stack_set_hhomogeneous()`.

Parameters

stack	a GtkStack

Returns

whether `stack` is horizontally homogeneous.

Since: 3.16

---

gtk_stack_set_vhomogeneous ()

```
void
gtk_stack_set_vhomogeneous (GtkStack *stack,
                            gboolean vhomogeneous);
```

Sets the GtkStack to be vertically homogeneous or not. If it is homogeneous, the GtkStack will request the same height for all its children. If it isn't, the stack may change height when a different child becomes visible.

Parameters

stack	a GtkStack
vhomogeneous	`TRUE` to make `stack` vertically homogeneous

Since: 3.16

---

gtk_stack_get_vhomogeneous ()

```
gboolean
gtk_stack_get_vhomogeneous (GtkStack *stack);
```

Gets whether `stack` is vertically homogeneous. See `gtk_stack_set_vhomogeneous()`.

Parameters

stack	a GtkStack

Returns

whether `stack` is vertically homogeneous.

Since: 3.16

---

gtk_stack_set_transition_duration ()

```
void
gtk_stack_set_transition_duration (GtkStack *stack,
                                   guint duration);
```

Sets the duration that transitions between pages in `stack` will take.

Parameters

stack	a GtkStack
duration	the new duration, in milliseconds

Since: 3.10

---

gtk_stack_get_transition_duration ()

```
guint
gtk_stack_get_transition_duration (GtkStack *stack);
```

Returns the amount of time (in milliseconds) that transitions between pages in `stack` will take.

Parameters

stack	a GtkStack

Returns

the transition duration

Since: 3.10

---

gtk_stack_set_transition_type ()

```
void
gtk_stack_set_transition_type (GtkStack *stack,
                               GtkStackTransitionType transition);
```

Sets the type of animation that will be used for transitions between pages in `stack` . Available types include various kinds of fades and slides.

The transition type can be changed without problems at runtime, so it is possible to change the animation based on the page that is about to become current.

Parameters

stack	a GtkStack
transition	the new transition type

Since: 3.10

---

gtk_stack_get_transition_type ()

```
GtkStackTransitionType
gtk_stack_get_transition_type (GtkStack *stack);
```

Gets the type of animation that will be used for transitions between pages in `stack` .

Parameters

stack	a GtkStack

Returns

the current transition type of `stack`

Since: 3.10

---

gtk_stack_get_transition_running ()

```
gboolean
gtk_stack_get_transition_running (GtkStack *stack);
```

Returns whether the `stack` is currently in a transition from one page to another.

Parameters

stack	a GtkStack

Returns

`TRUE` if the transition is currently running, `FALSE` otherwise.

Since: 3.12

---

gtk_stack_get_interpolate_size ()

```
gboolean
gtk_stack_get_interpolate_size (GtkStack *stack);
```

Returns wether the GtkStack is set up to interpolate between the sizes of children on page switch.

Parameters

stack	A GtkStack

Returns

`TRUE` if child sizes are interpolated

Since: 3.18

---

gtk_stack_set_interpolate_size ()

```
void
gtk_stack_set_interpolate_size (GtkStack *stack,
                                gboolean interpolate_size);
```

Sets whether or not `stack` will interpolate its size when changing the visible child. If the “interpolate-size” property is set to `TRUE`, `stack` will interpolate its size between the current one and the one it'll take after changing the visible child, according to the set transition duration.

Parameters

stack	A GtkStack
interpolate_size	the new value

Since: 3.18

Types and Values

struct GtkStack

```
struct GtkStack;
```

---

enum GtkStackTransitionType

These enumeration values describe the possible transitions between pages in a GtkStack widget.

New values may be added to this enumeration over time.

Members

GTK_STACK_TRANSITION_TYPE_NONE	No transition
GTK_STACK_TRANSITION_TYPE_CROSSFADE	A cross-fade
GTK_STACK_TRANSITION_TYPE_SLIDE_RIGHT	Slide from left to right
GTK_STACK_TRANSITION_TYPE_SLIDE_LEFT	Slide from right to left
GTK_STACK_TRANSITION_TYPE_SLIDE_UP	Slide from bottom up
GTK_STACK_TRANSITION_TYPE_SLIDE_DOWN	Slide from top down
GTK_STACK_TRANSITION_TYPE_SLIDE_LEFT_RIGHT	Slide from left or right according to the children order
GTK_STACK_TRANSITION_TYPE_SLIDE_UP_DOWN	Slide from top down or bottom up according to the order
GTK_STACK_TRANSITION_TYPE_OVER_UP	Cover the old page by sliding up. Since 3.12
GTK_STACK_TRANSITION_TYPE_OVER_DOWN	Cover the old page by sliding down. Since: 3.12
GTK_STACK_TRANSITION_TYPE_OVER_LEFT	Cover the old page by sliding to the left. Since: 3.12
GTK_STACK_TRANSITION_TYPE_OVER_RIGHT	Cover the old page by sliding to the right. Since: 3.12
GTK_STACK_TRANSITION_TYPE_UNDER_UP	Uncover the new page by sliding up. Since 3.12
GTK_STACK_TRANSITION_TYPE_UNDER_DOWN	Uncover the new page by sliding down. Since: 3.12
GTK_STACK_TRANSITION_TYPE_UNDER_LEFT	Uncover the new page by sliding to the left. Since: 3.12
GTK_STACK_TRANSITION_TYPE_UNDER_RIGHT	Uncover the new page by sliding to the right. Since: 3.12
GTK_STACK_TRANSITION_TYPE_OVER_UP_DOWN	Cover the old page sliding up or uncover the new page sliding down, according to order. Since: 3.12
GTK_STACK_TRANSITION_TYPE_OVER_DOWN_UP	Cover the old page sliding down or uncover the new page sliding up, according to order. Since: 3.14
GTK_STACK_TRANSITION_TYPE_OVER_LEFT_RIGHT	Cover the old page sliding left or uncover the

	new page sliding right, according to order. Since: 3.14
GTK_STACK_TRANSITION_TYPE_OVER_RIGHT_LEFT	Cover the old page sliding right or uncover the new page sliding left, according to order. Since: 3.14

Property Details

The “hhomogeneous” property

“hhomogeneous” gboolean

TRUE if the stack allocates the same width for all children.

Flags: Read / Write

Default value: TRUE

Since: 3.16

The “homogeneous” property

“homogeneous” gboolean

Homogeneous sizing.

Flags: Read / Write

Default value: TRUE

The “interpolate-size” property

“interpolate-size” gboolean

Whether or not the size should smoothly change when changing between differently sized children.

Flags: Read / Write

Default value: FALSE

The “transition-duration” property

“transition-duration” guint

The animation duration, in milliseconds.

Flags: Read / Write

Default value: 200

The “transition-running” property

“transition-running” gboolean

Whether or not the transition is currently running.

Flags: Read

Default value: FALSE

The “transition-type” property

“transition-type” GtkStackTransitionType

The type of animation used to transition.

Flags: Read / Write

Default value: GTK_STACK_TRANSITION_TYPE_NONE

The “vhomogeneous” property

“vhomogeneous” gboolean

TRUE if the stack allocates the same height for all children.

Flags: Read / Write

Default value: TRUE

Since: 3.16

The “visible-child” property

“visible-child” GtkWidget *

The widget currently visible in the stack.

Flags: Read / Write

The “visible-child-name” property

“visible-child-name” gchar *

The name of the widget currently visible in the stack.

Flags: Read / Write

Default value: NULL

Child Property Details

The “icon-name” child property

“icon-name” gchar *

The icon name of the child page.

Flags: Read / Write

Default value: NULL

The “name” child property

“name” gchar *

The name of the child page.

Flags: Read / Write

Default value: NULL

The “needs-attention” child property

“needs-attention” gboolean

Sets a flag specifying whether the child requires the user attention. This is used by the GtkStackSwitcher to change the appearance of the corresponding button when a page needs attention and it is not the current one.

Flags: Read / Write

Default value: FALSE

Since: 3.12

The “position” child property

“position” gint

The index of the child in the parent.

Flags: Read / Write

Allowed values: >= -1

Default value: 0

The “title” child property

“title” gchar *

The title of the child page.

Flags: Read / Write

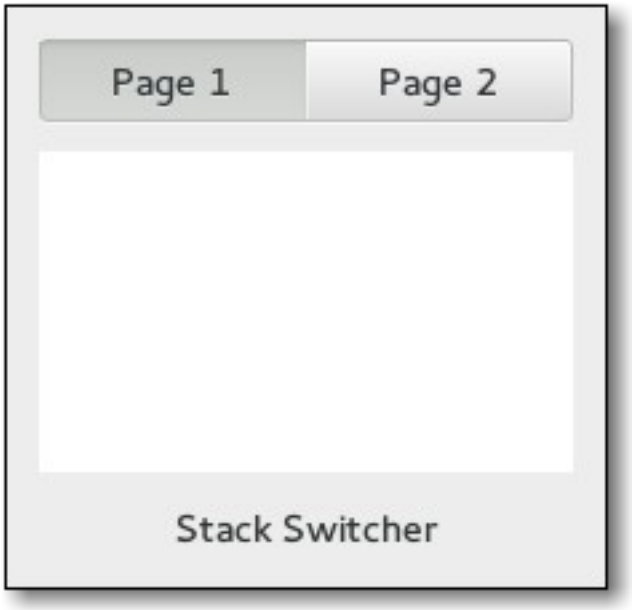
Default value: NULL

See Also

GtkNotebook, GtkStackSwitcher

GtkStackSwitcher

GtkStackSwitcher — A controller for GtkStack



Functions

GtkWidget *	gtk_stack_switcher_new ()
void	gtk_stack_switcher_set_stack ()
GtkStack *	gtk_stack_switcher_get_stack ()

Properties

gint	icon-size	Read / Write
GtkStack *	stack	Read / Write / Construct

Types and Values

struct	GtkStackSwitcher

Object Hierarchy

```
GObject
```

```
    └── GInitiallyUnowned
        └── GtkWidget
            └── GtkContainer
                └── GtkBox
                    └── GtkStackSwitcher
```

Implemented Interfaces

GtkStackSwitcher implements AtkImplementorIface, GtkBuildable and GtkOrientable.

Includes

```
#include <gtk/gtk.h>
```

Description

The GtkStackSwitcher widget acts as a controller for a GtkStack; it shows a row of buttons to switch between the various pages of the associated stack widget.

All the content for the buttons comes from the child properties of the GtkStack; the button visibility in a GtkStackSwitcher widget is controlled by the visibility of the child in the GtkStack.

It is possible to associate multiple GtkStackSwitcher widgets with the same GtkStack widget.

The GtkStackSwitcher widget was added in 3.10.

CSS nodes

GtkStackSwitcher has a single CSS node named stackswitcher and style class .stack-switcher.

When circumstances require it, GtkStackSwitcher adds the .needs-attention style class to the widgets representing the stack pages.

Functions

gtk_stack_switcher_new ()

```
GtkWidget *
gtk_stack_switcher_new (void);
```

Create a new GtkStackSwitcher.

Returns

a new GtkStackSwitcher.

Since: 3.10

---

gtk_stack_switcher_set_stack ()

```
void
gtk_stack_switcher_set_stack (GtkStackSwitcher *switcher,
                              GtkStack *stack);
```

Sets the stack to control.

Parameters

switcher	a GtkStackSwitcher	
stack	a GtkStack.	[allow-none]

Since: 3.10

---

gtk_stack_switcher_get_stack ()

```
GtkStack *
gtk_stack_switcher_get_stack (GtkStackSwitcher *switcher);
```

Retrieves the stack. See `gtk_stack_switcher_set_stack()`.

Parameters

switcher	a GtkStackSwitcher

Returns

the stack, or `NULL` if none has been set explicitly.

[nullable][transfer none]

Since: 3.10

Types and Values

struct GtkStackSwitcher

```
struct GtkStackSwitcher;
```

Property Details

The “icon-size” property

```
  “icon-size”                gint
```

Use the "icon-size" property to change the size of the image displayed when a GtkStackSwitcher is displaying icons.

Flags: Read / Write

Allowed values: >= 0

Default value: 1

Since: 3.20

The “stack” property

“stack” GtkStack *

Stack.

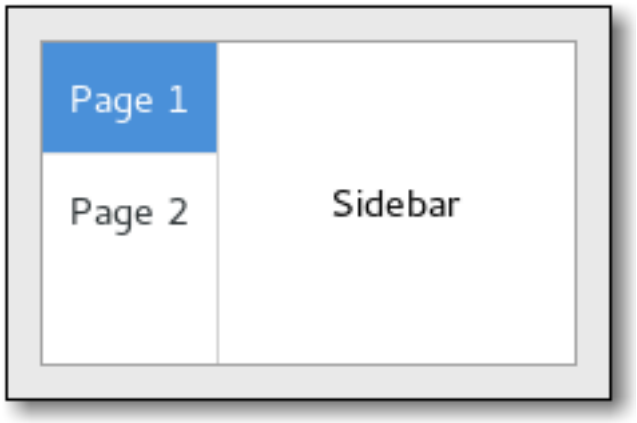
Flags: Read / Write / Construct

See Also

GtkStack

GtkStackSidebar

GtkStackSidebar — An automatic sidebar widget



Functions

GtkWidget *	gtk_stack_sidebar_new ()
void	gtk_stack_sidebar_set_stack ()
GtkStack *	gtk_stack_sidebar_get_stack ()

Properties

GtkStack *	stack	Read / Write

Types and Values

struct	GtkStackSidebar

struct GtkStackSidebarClass

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkContainer
            └── GtkBin
                └── GtkStackSidebar
```

Implemented Interfaces

GtkStackSidebar implements AtkImplementorIface and GtkBuildable.

Includes

```
#include <gtk/gtk.h>
```

Description

A GtkStackSidebar enables you to quickly and easily provide a consistent "sidebar" object for your user interface.

In order to use a GtkStackSidebar, you simply use a GtkStack to organize your UI flow, and add the sidebar to your sidebar area. You can use `gtk_stack_sidebar_set_stack()` to connect the GtkStackSidebar to the GtkStack.

CSS nodes

GtkStackSidebar has a single CSS node with name stacksidebar and style class .sidebar.

When circumstances require it, GtkStackSidebar adds the .needs-attention style class to the widgets representing the stack pages.

Functions

gtk_stack_sidebar_new ()

```
GtkWidget *
gtk_stack_sidebar_new (void);
```

Creates a new sidebar.

Returns

the new GtkStackSidebar

Since: 3.16

gtk_stack_sidebar_set_stack ()

```
void
gtk_stack_sidebar_set_stack (GtkStackSidebar *sidebar,
                             GtkStack *stack);
```

Set the GtkStack associated with this GtkStackSidebar.

The sidebar widget will automatically update according to the order (packing) and items within the given GtkStack.

Parameters

sidebar	a GtkStackSidebar
stack	a GtkStack

Since: 3.16

---

gtk_stack_sidebar_get_stack ()

```
GtkStack *
gtk_stack_sidebar_get_stack (GtkStackSidebar *sidebar);
```

Retrieves the stack. See `gtk_stack_sidebar_set_stack()`.

Parameters

sidebar	a GtkStackSidebar

Returns

the associated GtkStack or `NULL` if none has been set explicitly.

[nullable][transfer none]

Since: 3.16

Types and Values

struct GtkStackSidebar

```
struct GtkStackSidebar;
```

---

struct GtkStackSidebarClass

```
struct GtkStackSidebarClass {
  GtkBinClass parent_class;

  /* Padding for future expansion */
  void (*_gtk_reserved1) (void);
  void (*_gtk_reserved2) (void);
  void (*_gtk_reserved3) (void);
  void (*_gtk_reserved4) (void);
};
```

Property Details

The “stack” property

```
  “stack”                    GtkStack *
```

Associated stack for this GtkStackSidebar.

Flags: Read / Write

GtkActionBar

GtkActionBar — A full width bar for presenting contextual actions

Functions

GtkWidget *	gtk_action_bar_new ()
void	gtk_action_bar_pack_start ()
void	gtk_action_bar_pack_end ()
GtkWidget *	gtk_action_bar_get_center_widget ()
void	gtk_action_bar_set_center_widget ()

Child Properties

GtkPackType	pack-type	Read / Write
gint	position	Read / Write

Types and Values

struct	GtkActionBar

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkContainer
            └── GtkBin
                └── GtkActionBar
```

Implemented Interfaces

GtkActionBar implements AtkImplementorIface and GtkBuildable.

Includes

```
#include <gtk/gtk.h>
```

Description

GtkActionBar is designed to present contextual actions. It is expected to be displayed below the content and expand horizontally to fill the area.

It allows placing children at the start or the end. In addition, it contains an internal centered box which is centered with respect to the full width of the box, even if the children at either side take up different amounts of space.

CSS nodes

GtkActionBar has a single CSS node with name actionbar.

Functions

gtk_action_bar_new ()

```
GtkWidget *
gtk_action_bar_new (void);
```

Creates a new GtkActionBar widget.

Returns

a new GtkActionBar

Since: 3.12

---

gtk_action_bar_pack_start ()

```
void
gtk_action_bar_pack_start (GtkActionBar *action_bar,
                           GtkWidget *child);
```

Adds `child` to `action_bar` , packed with reference to the start of the `action_bar` .

Parameters

action_bar	A GtkActionBar
child	the GtkWidget to be added to `action_bar`

Since: 3.12

---

gtk_action_bar_pack_end ()

```
void
gtk_action_bar_pack_end (GtkActionBar *action_bar,
                         GtkWidget *child);
```

Adds `child` to `action_bar` , packed with reference to the end of the `action_bar` .

Parameters

action_bar	A GtkActionBar
child	the GtkWidget to be added to `action_bar`

Since: 3.12

---

gtk_action_bar_get_center_widget ()

```
GtkWidget *
gtk_action_bar_get_center_widget (GtkActionBar *action_bar);
```

Retrieves the center bar widget of the bar.

Parameters

action_bar	a GtkActionBar

Returns

the center GtkWidget or `NULL`.

[transfer none][nullable]

Since: 3.12

---

gtk_action_bar_set_center_widget ()

```
void
gtk_action_bar_set_center_widget (GtkActionBar *action_bar,
                                  GtkWidget *center_widget);
```

Sets the center widget for the GtkActionBar.

Parameters

action_bar	a GtkActionBar	
center_widget	a widget to use for the center.	[allow-none]

Since: 3.12

Types and Values

struct GtkActionBar

```
struct GtkActionBar;
```

Child Property Details

The “pack-type” child property

“pack-type”	GtkPackType

A GtkPackType indicating whether the child is packed with reference to the start or end of the parent.

Flags: Read / Write

Default value: GTK_PACK_START

The “position” child property

“position”	gint

The index of the child in the parent.

Flags: Read / Write

Allowed values: >= -1

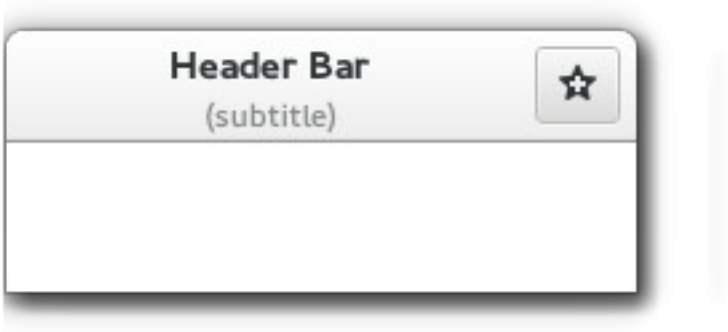
Default value: 0

See Also

GtkBox

GtkHeaderBar

GtkHeaderBar — A box with a centered child



Functions

GtkWidget *	gtk_header_bar_new ()
void	gtk_header_bar_set_title ()
const gchar *	gtk_header_bar_get_title ()
void	gtk_header_bar_set_subtitle ()
const gchar *	gtk_header_bar_get_subtitle ()
void	gtk_header_bar_set_has_subtitle ()
gboolean	gtk_header_bar_get_has_subtitle ()
void	gtk_header_bar_set_custom_title ()
GtkWidget *	gtk_header_bar_get_custom_title ()
void	gtk_header_bar_pack_start ()
void	gtk_header_bar_pack_end ()
void	gtk_header_bar_set_show_close_button ()
gboolean	gtk_header_bar_get_show_close_button ()
void	gtk_header_bar_set_decoration_layout ()
const gchar *	gtk_header_bar_get_decoration_layout ()

Properties

GtkWidget *	custom-title	Read / Write
gchar *	decoration-layout	Read / Write
gboolean	decoration-layout-set	Read / Write
gboolean	has-subtitle	Read / Write
gboolean	show-close-button	Read / Write
gint	spacing	Read / Write
gchar *	subtitle	Read / Write
gchar *	title	Read / Write

Child Properties

GtkPackType	pack-type	Read / Write
gint	position	Read / Write

Types and Values

struct	GtkHeaderBar

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkContainer
            └── GtkHeaderBar
```

Implemented Interfaces

GtkHeaderBar implements AtkImplementorIface and GtkBuildable.

Includes

```
#include <gtk/gtk.h>
```

Description

GtkHeaderBar is similar to a horizontal GtkBox. It allows children to be placed at the start or the end. In addition, it allows a title and subtitle to be displayed. The title will be centered with respect to the width of the box, even if the children at either side take up different amounts of space. The height of the titlebar will be set to provide sufficient space for the subtitle, even if none is currently set. If a subtitle is not needed, the space reservation can be turned off with `gtk_header_bar_set_has_subtitle()`.

GtkHeaderBar can add typical window frame controls, such as minimize, maximize and close buttons, or the window icon.

For these reasons, GtkHeaderBar is the natural choice for use as the custom titlebar widget of a GtkWindow (see `gtk_window_set_titlebar()`), as it gives features typical of titlebars while allowing the addition of child widgets.

Functions

gtk_header_bar_new ()

```
GtkWidget *
gtk_header_bar_new (void);
```

Creates a new GtkHeaderBar widget.

Returns

a new GtkHeaderBar

Since: 3.10

gtk_header_bar_set_title ()

```
void
gtk_header_bar_set_title (GtkHeaderBar *bar,
                          const gchar *title);
```

Sets the title of the GtkHeaderBar. The title should help a user identify the current view. A good title should not include the application name.

Parameters

bar	a GtkHeaderBar	
title	a title, or `NULL`.	[allow-none]

Since: 3.10

---

gtk_header_bar_get_title ()

```
const gchar *
gtk_header_bar_get_title (GtkHeaderBar *bar);
```

Retrieves the title of the header. See `gtk_header_bar_set_title()`.

Parameters

bar	a GtkHeaderBar

Returns

the title of the header, or `NULL` if none has been set explicitly. The returned string is owned by the widget and must not be modified or freed.

[nullable]

Since: 3.10

---

gtk_header_bar_set_subtitle ()

```
void
gtk_header_bar_set_subtitle (GtkHeaderBar *bar,
                             const gchar *subtitle);
```

Sets the subtitle of the GtkHeaderBar. The title should give a user an additional detail to help him identify the current view.

Note that GtkHeaderBar by default reserves room for the subtitle, even if none is currently set. If this is not desired, set the “has-subtitle” property to `FALSE`.

Parameters

bar	a GtkHeaderBar	
subtitle	a subtitle, or `NULL`.	[allow-none]

Since: 3.10

---

gtk_header_bar_get_subtitle ()

```
const gchar *
gtk_header_bar_get_subtitle (GtkHeaderBar *bar);
```

Retrieves the subtitle of the header. See `gtk_header_bar_set_subtitle()`.

Parameters

bar	a GtkHeaderBar

Returns

the subtitle of the header, or `NULL` if none has been set explicitly. The returned string is owned by the widget and must not be modified or freed.

[nullable]

Since: 3.10

---

gtk_header_bar_set_has_subtitle ()

```
void
gtk_header_bar_set_has_subtitle (GtkHeaderBar *bar,
                                 gboolean setting);
```

Sets whether the header bar should reserve space for a subtitle, even if none is currently set.

Parameters

bar	a GtkHeaderBar
setting	`TRUE` to reserve space for a subtitle

Since: 3.12

---

gtk_header_bar_get_has_subtitle ()

```
gboolean
gtk_header_bar_get_has_subtitle (GtkHeaderBar *bar);
```

Retrieves whether the header bar reserves space for a subtitle, regardless if one is currently set or not.

Parameters

bar	a GtkHeaderBar

Returns

`TRUE` if the header bar reserves space for a subtitle

Since: 3.12

---

gtk_header_bar_set_custom_title ()

```
void
gtk_header_bar_set_custom_title (GtkHeaderBar *bar,
                                 GtkWidget *title_widget);
```

Sets a custom title for the GtkHeaderBar.

The title should help a user identify the current view. This supersedes any title set by `gtk_header_bar_set_title()` or `gtk_header_bar_set_subtitle()`. To achieve the same style as the builtin title and subtitle, use the “title” and “subtitle” style classes.

You should set the custom title to `NULL`, for the header title label to be visible again.

Parameters

bar	a GtkHeaderBar	
title_widget	a custom widget to use for a title.	[allow-none]

Since: 3.10

---

gtk_header_bar_get_custom_title ()

```
GtkWidget *
gtk_header_bar_get_custom_title (GtkHeaderBar *bar);
```

Retrieves the custom title widget of the header. See `gtk_header_bar_set_custom_title()`.

Parameters

bar	a GtkHeaderBar

Returns

the custom title widget of the header, or `NULL` if none has been set explicitly.

[nullable][transfer none]

Since: 3.10

---

gtk_header_bar_pack_start ()

```
void
gtk_header_bar_pack_start (GtkHeaderBar *bar,
                           GtkWidget *child);
```

Adds `child` to `bar` , packed with reference to the start of the `bar` .

Parameters

bar	A GtkHeaderBar
child	the GtkWidget to be added to `bar`

Since: 3.10

---

gtk_header_bar_pack_end ()

```
void
gtk_header_bar_pack_end (GtkHeaderBar *bar,
                         GtkWidget *child);
```

Adds `child` to `bar` , packed with reference to the end of the `bar` .

Parameters

bar	A GtkHeaderBar
child	the GtkWidget to be added to `bar`

Since: 3.10

---

gtk_header_bar_set_show_close_button ()

```
void
gtk_header_bar_set_show_close_button (GtkHeaderBar *bar,
                                      gboolean setting);
```

Sets whether this header bar shows the standard window decorations, including close, maximize, and minimize.

Parameters

bar	a GtkHeaderBar
setting	`TRUE` to show standard window decorations

Since: 3.10

---

gtk_header_bar_get_show_close_button ()

```
gboolean
gtk_header_bar_get_show_close_button (GtkHeaderBar *bar);
```

Returns whether this header bar shows the standard window decorations.

Parameters

bar	a GtkHeaderBar

Returns

TRUE if the decorations are shown

Since: 3.10

---

gtk_header_bar_set_decoration_layout ()

```
void
gtk_header_bar_set_decoration_layout (GtkHeaderBar *bar,
                                      const gchar *layout);
```

Sets the decoration layout for this header bar, overriding the “gtk-decoration-layout” setting.

There can be valid reasons for overriding the setting, such as a header bar design that does not allow for buttons to take room on the right, or only offers room for a single close button. Split header bars are another example for overriding the setting.

The format of the string is button names, separated by commas. A colon separates the buttons that should appear on the left from those on the right. Recognized button names are minimize, maximize, close, icon (the window icon) and menu (a menu button for the fallback app menu).

For example, “menu:minimize,maximize,close” specifies a menu on the left, and minimize, maximize and close buttons on the right.

Parameters

bar	a GtkHeaderBar
layout	a decoration layout, or NULL to unset the layout. [allow-none]

Since: 3.12

---

gtk_header_bar_get_decoration_layout ()

```
const gchar *
gtk_header_bar_get_decoration_layout (GtkHeaderBar *bar);
```

Gets the decoration layout set with gtk_header_bar_set_decoration_layout().

Parameters

bar	a GtkHeaderBar

Returns

the decoration layout

Since: 3.12

Types and Values

struct GtkHeaderBar

```
struct GtkHeaderBar;
```

Property Details

The “custom-title” property

```
  “custom-title”             GtkWidget *
```

Custom title widget to display.

Flags: Read / Write

The “decoration-layout” property

```
  “decoration-layout”        gchar *
```

The decoration layout for buttons. If this property is not set, the “gtk-decoration-layout” setting is used.

See `gtk_header_bar_set_decoration_layout()` for information about the format of this string.

Flags: Read / Write

Default value: NULL

Since: 3.12

The “decoration-layout-set” property

```
  “decoration-layout-set”    gboolean
```

Set to `TRUE` if “decoration-layout” is set.

Flags: Read / Write

Default value: FALSE

Since: 3.12

The “has-subtitle” property

“has-subtitle” gboolean

If TRUE, reserve space for a subtitle, even if none is currently set.

Flags: Read / Write

Default value: TRUE

Since: 3.12

The “show-close-button” property

“show-close-button” gboolean

Whether to show window decorations.

Which buttons are actually shown and where is determined by the “decoration-layout” property, and by the state of the window (e.g. a close button will not be shown if the window can't be closed).

Flags: Read / Write

Default value: FALSE

The “spacing” property

“spacing” gint

The amount of space between children.

Flags: Read / Write

Allowed values: >= 0

Default value: 6

The “subtitle” property

“subtitle” gchar *

The subtitle to display.

Flags: Read / Write

Default value: NULL

The “title” property

“title” gchar *

The title to display.

Flags: Read / Write

Default value: NULL

Child Property Details

The “pack-type” child property

“pack-type” GtkPackType

A GtkPackType indicating whether the child is packed with reference to the start or end of the parent.

Flags: Read / Write

Default value: GTK_PACK_START

The “position” child property

“position” gint

The index of the child in the parent.

Flags: Read / Write

Allowed values: >= -1

Default value: 0

See Also

GtkBox, GtkActionBar

GtkOverlay

GtkOverlay — A container which overlays widgets on top of each other

Functions

GtkWidget *	gtk_overlay_new ()
void	gtk_overlay_add_overlay ()
void	gtk_overlay_reorder_overlay ()
gboolean	gtk_overlay_get_overlay_pass_through ()
void	gtk_overlay_set_overlay_pass_through ()

Child Properties

gint	index	Read / Write
gboolean	pass-through	Read / Write

Signals

gboolean	get-child-position	Run Last

Types and Values

struct	GtkOverlay
struct	GtkOverlayClass

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkContainer
            └── GtkBin
                └── GtkOverlay
```

Implemented Interfaces

GtkOverlay implements AtkImplementorIface and GtkBuildable.

Includes

```
#include <gtk/gtk.h>
```

Description

GtkOverlay is a container which contains a single main child, on top of which it can place “overlay” widgets. The position of each overlay widget is determined by its “halign” and “valign” properties. E.g. a widget with both alignments set to `GTK_ALIGN_START` will be placed at the top left corner of the GtkOverlay container, whereas an overlay with halign set to `GTK_ALIGN_CENTER` and valign set to `GTK_ALIGN_END` will be placed a the bottom edge of the GtkOverlay, horizontally centered. The position can be adjusted by setting the margin properties of the child to non-zero values.

More complicated placement of overlays is possible by connecting to the “get-child-position” signal.

An overlay’s minimum and natural sizes are those of its main child. The sizes of overlay children are not considered when measuring these preferred sizes.

GtkOverlay as GtkBuildable

The GtkOverlay implementation of the GtkBuildable interface supports placing a child as an overlay by specifying “overlay” as the “type” attribute of a `<child>` element.

CSS nodes

GtkOverlay has a single CSS node with the name “overlay”. Overlay children whose alignments cause them to be positioned at an edge get the style classes “.left”, “.right”, “.top”, and/or “.bottom” according to their position.

Functions

gtk_overlay_new ()

```
GtkWidget *
gtk_overlay_new (void);
```

Creates a new GtkOverlay.

Returns

a new GtkOverlay object.

Since: 3.2

---

gtk_overlay_add_overlay ()

```
void
gtk_overlay_add_overlay (GtkOverlay *overlay,
                         GtkWidget *widget);
```

Adds `widget` to `overlay` .

The widget will be stacked on top of the main widget added with `gtk_container_add()`.

The position at which `widget` is placed is determined from its “halign” and “valign” properties.

Parameters

overlay	a GtkOverlay
widget	a GtkWidget to be added to the container

Since: 3.2

---

gtk_overlay_reorder_overlay ()

```
void
gtk_overlay_reorder_overlay (GtkOverlay *overlay,
                             GtkWidget *child,
                             int index_);
```

Moves `child` to a new `index` in the list of `overlay` children. The list contains overlays in the order that these were added to `overlay` by default. See also “index”.

A widget’s index in the `overlay` children list determines which order the children are drawn if they overlap. The first child is drawn at the bottom. It also affects the default focus chain order.

Parameters

overlay	a GtkOverlay
child	the overlaid GtkWidget to move
index_	the new index for `child` in the list of overlay children of `overlay` ,

	starting from 0. If negative, indicates the end of the list

Since: 3.18

---

gtk_overlay_get_overlay_pass_through ()

```
gboolean
gtk_overlay_get_overlay_pass_through (GtkOverlay *overlay,
                                      GtkWidget *widget);
```

Convenience function to get the value of the “pass-through” child property for `widget` .

Parameters

overlay	a GtkOverlay
widget	an overlay child of GtkOverlay

Returns

whether the widget is a pass through child.

Since: 3.18

---

gtk_overlay_set_overlay_pass_through ()

```
void
gtk_overlay_set_overlay_pass_through (GtkOverlay *overlay,
                                      GtkWidget *widget,
                                      gboolean pass_through);
```

Convenience function to set the value of the “pass-through” child property for `widget` .

Parameters

overlay	a GtkOverlay
widget	an overlay child of GtkOverlay
pass_through	whether the child should pass the input through

Since: 3.18

Types and Values

struct GtkOverlay

```
struct GtkOverlay;
```

---

struct GtkOverlayClass

```
struct GtkOverlayClass {
  GtkBinClass parent_class;


  gboolean (*get_child_position) (GtkOverlay    *overlay,
                                  GtkWidget     *widget,
                                  GtkAllocation *allocation);
};
```

Members

`get_child_position` ()	Signal emitted to determine the position and size of any overlay child widgets.

Child Property Details

The `"index"` child property

`"index"` `gint`

The index of the overlay child in the parent (or -1 for the main child). See `gtk_overlay_reorder_overlay()`.

Flags: Read / Write

Allowed values: >= -1

Default value: 0

Since: 3.18

The `"pass-through"` child property

`"pass-through"` `gboolean`

Whether to pass input through the overlay child to the main child. (Of course, this has no effect when set on the main child itself.)

Flags: Read / Write

Default value: FALSE

Since: 3.18

Signal Details

The "get-child-position" signal

```
gboolean
user_function (GtkOverlay   *overlay,
               GtkWidget    *widget,
               GdkRectangle *allocation,
               gpointer      user_data)
```

The ::get-child-position signal is emitted to determine the position and size of any overlay child widgets. A handler for this signal should fill `allocation` with the desired position and size for `widget` , relative to the 'main' child of `overlay` .

The default handler for this signal uses the `widget` 's halign and valign properties to determine the position and gives the widget its natural size (except that an alignment of `GTK_ALIGN_FILL` will cause the overlay to be full-width/height). If the main child is a GtkScrolledWindow, the overlays are placed relative to its contents.

Parameters

overlay	the GtkOverlay	
widget	the child widget to position	
allocation	return location for the allocation.	[type Gdk.Rectangle][out caller-allocates]
user_data	user data set when the signal handler was connected.	

Returns

`TRUE` if the `allocation` has been filled

Flags: Run Last

GtkButtonBox

GtkButtonBox — A container for arranging buttons

Functions

GtkWidget *	gtk_button_box_new ()
GtkButtonBoxStyle	gtk_button_box_get_layout ()
gboolean	gtk_button_box_get_child_secondary ()
gboolean	gtk_button_box_get_child_non_homogeneous ()
void	gtk_button_box_set_layout ()
void	gtk_button_box_set_child_secondary ()
void	gtk_button_box_set_child_non_homogeneous ()

Properties

GtkButtonBoxStyle	layout-style	Read / Write

Child Properties

gboolean	non-homogeneous	Read / Write
gboolean	secondary	Read / Write

Style Properties

gint	child-internal-pad-x	Read
gint	child-internal-pad-y	Read
gint	child-min-height	Read
gint	child-min-width	Read

Types and Values

struct	GtkButtonBox
struct	GtkButtonBoxClass
enum	GtkButtonBoxStyle

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkContainer
            └── GtkBox
                └── GtkButtonBox
                    ├── GtkHButtonBox
                    └── GtkVButtonBox
```

Implemented Interfaces

GtkButtonBox implements AtkImplementorIface, GtkBuildable and GtkOrientable.

Includes

```
#include <gtk/gtk.h>
```

Description

A button box should be used to provide a consistent layout of buttons throughout your application. The layout/spacing can be altered by the programmer, or if desired, by the user to alter the “feel” of a program to a small degree.

gtk_button_box_get_layout() and `gtk_button_box_set_layout()` retrieve and alter the method used to spread the buttons in a button box across the container, respectively.

The main purpose of GtkButtonBox is to make sure the children have all the same size. GtkButtonBox gives all

children the same size, but it does allow 'outliers' to keep their own larger size.

To exempt individual children from homogeneous sizing regardless of their 'outlier' status, you can set the non-homogeneous child property.

CSS nodes

GtkButtonBox uses a single CSS node with name buttonbox.

Functions

gtk_button_box_new ()

```
GtkWidget *
gtk_button_box_new (GtkOrientation orientation);
```

Creates a new GtkButtonBox.

Parameters

orientation	the box's orientation.

Returns

a new GtkButtonBox.

Since: 3.0

---

gtk_button_box_get_layout ()

```
GtkButtonBoxStyle
gtk_button_box_get_layout (GtkButtonBox *widget);
```

Retrieves the method being used to arrange the buttons in a button box.

Parameters

widget	a GtkButtonBox

Returns

the method used to lay out buttons in `widget`.

---

gtk_button_box_get_child_secondary ()

```
gboolean
gtk_button_box_get_child_secondary (GtkButtonBox *widget,
                                    GtkWidget *child);
```

Returns whether `child` should appear in a secondary group of children.

Parameters

widget	a GtkButtonBox
child	a child of `widget`

Returns

whether `child` should appear in a secondary group of children.

Since: 2.4

---

gtk_button_box_get_child_non_homogeneous ()

```
gboolean
gtk_button_box_get_child_non_homogeneous
                                (GtkButtonBox *widget,
                                 GtkWidget *child);
```

Returns whether the child is exempted from homogenous sizing.

Parameters

widget	a GtkButtonBox
child	a child of `widget`

Returns

`TRUE` if the child is not subject to homogenous sizing

Since: 3.2

---

gtk_button_box_set_layout ()

```
void
gtk_button_box_set_layout (GtkButtonBox *widget,
                           GtkButtonBoxStyle layout_style);
```

Changes the way buttons are arranged in their container.

Parameters

widget	a GtkButtonBox

layout_style	the new layout style

gtk_button_box_set_child_secondary ()

```
void
gtk_button_box_set_child_secondary (GtkButtonBox *widget,
                                    GtkWidget *child,
                                    gboolean is_secondary);
```

Sets whether `child` should appear in a secondary group of children. A typical use of a secondary child is the help button in a dialog.

This group appears after the other children if the style is `GTK_BUTTONBOX_START`, `GTK_BUTTONBOX_SPREAD` or `GTK_BUTTONBOX_EDGE`, and before the other children if the style is `GTK_BUTTONBOX_END`. For horizontal button boxes, the definition of before/after depends on direction of the widget (see `gtk_widget_set_direction()`). If the style is `GTK_BUTTONBOX_START` or `GTK_BUTTONBOX_END`, then the secondary children are aligned at the other end of the button box from the main children. For the other styles, they appear immediately next to the main children.

Parameters

widget	a GtkButtonBox
child	a child of `widget`
is_secondary	if `TRUE`, the `child` appears in a secondary group of the button box.

gtk_button_box_set_child_non_homogeneous ()

```
void
gtk_button_box_set_child_non_homogeneous
                               (GtkButtonBox *widget,
                                GtkWidget *child,
                                gboolean non_homogeneous);
```

Sets whether the child is exempted from homogeous sizing.

Parameters

widget	a GtkButtonBox
child	a child of `widget`
non_homogeneous	the new value

Since: 3.2

Types and Values

struct GtkButtonBox

```
struct GtkButtonBox;
```

---

struct GtkButtonBoxClass

```
struct GtkButtonBoxClass {
  GtkBoxClass parent_class;
};
```

Members

---

enum GtkButtonBoxStyle

Used to dictate the style that a GtkButtonBox uses to layout the buttons it contains.

Members

GTK_BUTTONBOX_SPREAD	Buttons are evenly spread across the box.
GTK_BUTTONBOX_EDGE	Buttons are placed at the edges of the box.
GTK_BUTTONBOX_START	Buttons are grouped towards the start of the box, (on the left for a HBox, or the top for a VBox).
GTK_BUTTONBOX_END	Buttons are grouped towards the end of the box, (on the right for a HBox, or the bottom for a VBox).
GTK_BUTTONBOX_CENTER	Buttons are centered in the box. Since 2.12.
GTK_BUTTONBOX_EXPAND	Buttons expand to fill the box. This entails giving buttons a "linked" appearance, making button sizes homogeneous, and setting spacing to 0 (same as calling `gtk_box_set_homogeneous()` and `gtk_box_set_spacing()` manually). Since 3.12.

Property Details

The `“layout-style”` property

```
  “layout-style”              GtkButtonBoxStyle
```

How to lay out the buttons in the box. Possible values are: spread, edge, start and end.

Flags: Read / Write

Default value: GTK_BUTTONBOX_EDGE

Child Property Details

The `"non-homogeneous"` child property

`"non-homogeneous"` `gboolean`

If TRUE, the child will not be subject to homogeneous sizing.

Flags: Read / Write

Default value: FALSE

The `"secondary"` child property

`"secondary"` `gboolean`

If TRUE, the child appears in a secondary group of children, suitable for, e.g., help buttons.

Flags: Read / Write

Default value: FALSE

Style Property Details

The `"child-internal-pad-x"` style property

`"child-internal-pad-x"` `gint`

The amount to increase a child's size on either side.

`GtkButtonBox:child-internal-pad-x` has been deprecated since version 3.20 and should not be used in newly-written code.

Use CSS padding instead.

Flags: Read

Allowed values: >= 0

Default value: 4

The `"child-internal-pad-y"` style property

`"child-internal-pad-y"` `gint`

The amount to increase a child's size on the top and bottom.

`GtkButtonBox:child-internal-pad-y` has been deprecated since version 3.20 and should not be used in newly-written code.

Use CSS padding instead.

Flags: Read

Allowed values: >= 0

Default value: 0

The “child-min-height” style property

`“child-min-height” gint`

The minimum height of buttons inside the box.

`GtkButtonBox:child-min-height` has been deprecated since version 3.20 and should not be used in newly-written code.

Use CSS min-height instead.

Flags: Read

Allowed values: >= 0

Default value: 27

The “child-min-width” style property

`“child-min-width” gint`

The minimum width of buttons inside the box.

`GtkButtonBox:child-min-width` has been deprecated since version 3.20 and should not be used in newly-written code.

Use CSS min-width instead.

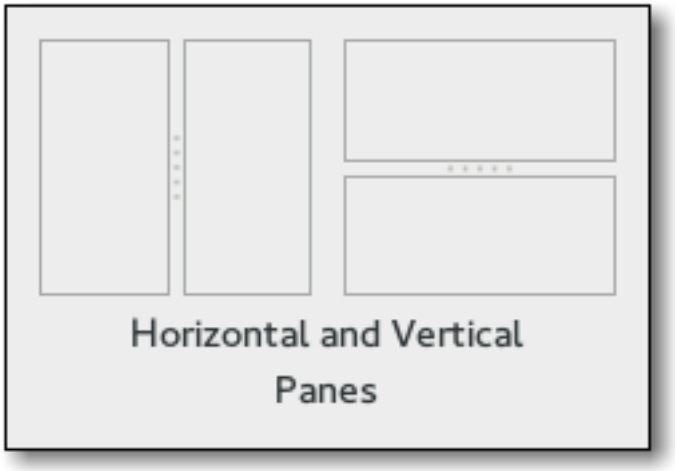
Flags: Read

Allowed values: >= 0

Default value: 85

GtkPaned

GtkPaned — A widget with two adjustable panes



Functions

GtkWidget *	gtk_paned_new ()
void	gtk_paned_add1 ()
void	gtk_paned_add2 ()
void	gtk_paned_pack1 ()
void	gtk_paned_pack2 ()
GtkWidget *	gtk_paned_get_child1 ()
GtkWidget *	gtk_paned_get_child2 ()
void	gtk_paned_set_position ()
gint	gtk_paned_get_position ()
GdkWindow *	gtk_paned_get_handle_window ()
void	gtk_paned_set_wide_handle ()
gboolean	gtk_paned_get_wide_handle ()

Properties

gint	max-position	Read
gint	min-position	Read
gint	position	Read / Write
gboolean	position-set	Read / Write
gboolean	wide-handle	Read / Write

Child Properties

gboolean	resize	Read / Write
gboolean	shrink	Read / Write

Style Properties

gint	handle-size	Read

Signals

gboolean	accept-position	Action
gboolean	cancel-position	Action
gboolean	cycle-child-focus	Action
gboolean	cycle-handle-focus	Action
gboolean	move-handle	Action
gboolean	toggle-handle-focus	Action

Types and Values

struct	GtkPaned

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkContainer
            └── GtkPaned
                ├── GtkHPaned
                └── GtkVPaned
```

Implemented Interfaces

GtkPaned implements AtkImplementorIface, GtkBuildable and GtkOrientable.

Includes

```
#include <gtk/gtk.h>
```

Description

GtkPaned has two panes, arranged either horizontally or vertically. The division between the two panes is adjustable by the user by dragging a handle.

Child widgets are added to the panes of the widget with `gtk_paned_pack1()` and `gtk_paned_pack2()`. The division between the two children is set by default from the size requests of the children, but it can be adjusted by the user.

A paned widget draws a separator between the two child widgets and a small handle that the user can drag to adjust the division. It does not draw any relief around the children or around the separator. (The space in which the separator is called the gutter.) Often, it is useful to put each child inside a GtkFrame with the shadow type set to `GTK_SHADOW_IN` so that the gutter appears as a ridge. No separator is drawn if one of the children is missing.

Each child has two options that can be set, `resize` and `shrink` . If `resize` is true, then when the GtkPaned is resized, that child will expand or shrink along with the paned widget. If `shrink` is true, then that child can be made smaller than its requisition by the user. Setting `shrink` to `FALSE` allows the application to set a minimum size. If `resize` is false for both children, then this is treated as if `resize` is true for both children.

The application can set the position of the slider as if it were set by the user, by calling

gtk_paned_set_position().

CSS nodes

```
paned
├── <child>
├── separator[.wide]
└── <child>
```

GtkPaned has a main CSS node with name paned, and a subnode for the separator with name separator. The subnode gets a .wide style class when the paned is supposed to be wide.

In horizontal orientation, the nodes of the children are always arranged from left to right. So :first-child will always select the leftmost child, regardless of text direction.

Creating a paned widget with minimum sizes.

```
GtkWidget *hpaned = gtk_paned_new (GTK_ORIENTATION_HORIZONTAL);
GtkWidget *frame1 = gtk_frame_new (NULL);
GtkWidget *frame2 = gtk_frame_new (NULL);
gtk_frame_set_shadow_type (GTK_FRAME (frame1), GTK_SHADOW_IN);
gtk_frame_set_shadow_type (GTK_FRAME (frame2), GTK_SHADOW_IN);

gtk_widget_set_size_request (hpaned, 200, -1);

gtk_paned_pack1 (GTK_PANED (hpaned), frame1, TRUE, FALSE);
gtk_widget_set_size_request (frame1, 50, -1);

gtk_paned_pack2 (GTK_PANED (hpaned), frame2, FALSE, FALSE);
gtk_widget_set_size_request (frame2, 50, -1);
```

Functions

gtk_paned_new ()

```
GtkWidget *
gtk_paned_new (GtkOrientation orientation);
```

Creates a new GtkPaned widget.

Parameters

orientation	the paned's orientation.

Returns

a new GtkPaned.

Since: 3.0

gtk_paned_add1 ()

```
void
gtk_paned_add1 (GtkPaned *paned,
                GtkWidget *child);
```

Adds a child to the top or left pane with default parameters. This is equivalent to `gtk_paned_pack1 (paned, child, FALSE, TRUE)`.

Parameters

paned	a paned widget
child	the child to add

---

gtk_paned_add2 ()

```
void
gtk_paned_add2 (GtkPaned *paned,
                GtkWidget *child);
```

Adds a child to the bottom or right pane with default parameters. This is equivalent to `gtk_paned_pack2 (paned, child, TRUE, TRUE)`.

Parameters

paned	a paned widget
child	the child to add

---

gtk_paned_pack1 ()

```
void
gtk_paned_pack1 (GtkPaned *paned,
                 GtkWidget *child,
                 gboolean resize,
                 gboolean shrink);
```

Adds a child to the top or left pane.

Parameters

paned	a paned widget
child	the child to add
resize	should this child expand when the paned widget is resized.
shrink	can this child be made smaller than its requisition.

---

gtk_paned_pack2 ()

```
void
gtk_paned_pack2 (GtkPaned *paned,
                 GtkWidget *child,
                 gboolean resize,
                 gboolean shrink);
```

Adds a child to the bottom or right pane.

Parameters

paned	a paned widget
child	the child to add
resize	should this child expand when the paned widget is resized.
shrink	can this child be made smaller than its requisition.

---

gtk_paned_get_child1 ()

```
GtkWidget *
gtk_paned_get_child1 (GtkPaned *paned);
```

Obtains the first child of the paned widget.

Parameters

paned	a GtkPaned widget

Returns

first child, or `NULL` if it is not set.

[nullable][transfer none]

Since: 2.4

---

gtk_paned_get_child2 ()

```
GtkWidget *
gtk_paned_get_child2 (GtkPaned *paned);
```

Obtains the second child of the paned widget.

Parameters

paned	a GtkPaned widget

Returns

second child, or `NULL` if it is not set.

[nullable][transfer none]

Since: 2.4

---

gtk_paned_set_position ()

```
void
gtk_paned_set_position (GtkPaned *paned,
                        gint position);
```

Sets the position of the divider between the two panes.

Parameters

paned	a GtkPaned widget
position	pixel position of divider, a negative value means that the position is unset.

---

gtk_paned_get_position ()

```
gint
gtk_paned_get_position (GtkPaned *paned);
```

Obtains the position of the divider between the two panes.

Parameters

paned	a GtkPaned widget

Returns

position of the divider

---

gtk_paned_get_handle_window ()

```
GdkWindow *
gtk_paned_get_handle_window (GtkPaned *paned);
```

Returns the GdkWindow of the handle. This function is useful when handling button or motion events because it enables the callback to distinguish between the window of the paned, a child and the handle.

Parameters

paned	a GtkPaned

Returns

the paned's handle window.

[transfer none]

Since: 2.20

---

gtk_paned_set_wide_handle ()

```
void
gtk_paned_set_wide_handle (GtkPaned *paned,
                           gboolean wide);
```

Sets the "wide-handle" property.

Parameters

paned	a GtkPaned
wide	the new value for the "wide-handle" property

Since: 3.16

---

gtk_paned_get_wide_handle ()

```
gboolean
gtk_paned_get_wide_handle (GtkPaned *paned);
```

Gets the "wide-handle" property.

Parameters

paned	a GtkPaned

Returns

`TRUE` if the paned should have a wide handle

Since: 3.16

Types and Values

struct GtkPaned

```
struct GtkPaned;
```

Property Details

The “max-position” property

```
  “max-position”              gint
```

The largest possible value for the position property. This property is derived from the size and shrinkability of the widget's children.

Flags: Read

Allowed values: >= 0

Default value: 2147483647

Since: 2.4

The “min-position” property

```
  “min-position”              gint
```

The smallest possible value for the position property. This property is derived from the size and shrinkability of the widget's children.

Flags: Read

Allowed values: >= 0

Default value: 0

Since: 2.4

The “position” property

```
  “position”                  gint
```

Position of paned separator in pixels (0 means all the way to the left/top).

Flags: Read / Write

Allowed values: >= 0

Default value: 0

The “position-set” property

```
  “position-set”              gboolean
```

TRUE if the Position property should be used.

Flags: Read / Write

Default value: FALSE

The “wide-handle” property

“wide-handle” gboolean

Setting this property to TRUE indicates that the paned needs to provide stronger visual separation (e.g. because it separates between two notebooks, whose tab rows would otherwise merge visually).

Flags: Read / Write

Default value: FALSE

Since: 3.16

Child Property Details

The “resize” child property

“resize” gboolean

The "resize" child property determines whether the child expands and shrinks along with the paned widget.

Flags: Read / Write

Default value: TRUE

Since: 2.4

The “shrink” child property

“shrink” gboolean

The "shrink" child property determines whether the child can be made smaller than its requisition.

Flags: Read / Write

Default value: TRUE

Since: 2.4

Style Property Details

The “handle-size” style property

“handle-size” gint

Width of handle.

Flags: Read

Allowed values: >= 0

Default value: 5

Signal Details

The `"accept-position"` signal

```
gboolean
user_function (GtkPaned *widget,
               gpointer  user_data)
```

The ::accept-position signal is a keybinding signal which gets emitted to accept the current position of the handle when moving it using key bindings.

The default binding for this signal is Return or Space.

Parameters

widget	the object that received the signal
user_data	user data set when the signal handler was connected.

Flags: Action

Since: 2.0

The `"cancel-position"` signal

```
gboolean
user_function (GtkPaned *widget,
               gpointer  user_data)
```

The ::cancel-position signal is a keybinding signal which gets emitted to cancel moving the position of the handle using key bindings. The position of the handle will be reset to the value prior to moving it.

The default binding for this signal is Escape.

Parameters

widget	the object that received the signal
user_data	user data set when the signal handler was connected.

Flags: Action

Since: 2.0

The “cycle-child-focus” signal

```
gboolean
user_function (GtkPaned *widget,
               gboolean  reversed,
               gpointer  user_data)
```

The ::cycle-child-focus signal is a keybinding signal which gets emitted to cycle the focus between the children of the paned.

The default binding is f6.

Parameters

widget	the object that received the signal
reversed	whether cycling backward or forward
user_data	user data set when the signal handler was connected.

Flags: Action

Since: 2.0

The “cycle-handle-focus” signal

```
gboolean
user_function (GtkPaned *widget,
               gboolean  reversed,
               gpointer  user_data)
```

The ::cycle-handle-focus signal is a keybinding signal which gets emitted to cycle whether the paned should grab focus to allow the user to change position of the handle by using key bindings.

The default binding for this signal is f8.

Parameters

widget	the object that received the signal
reversed	whether cycling backward or forward
user_data	user data set when the signal handler was connected.

Flags: Action

Since: 2.0

The “move-handle” signal

```
gboolean
user_function (GtkPaned     *widget,
               GtkScrollType scroll_type,
               gpointer      user_data)
```

The ::move-handle signal is a keybinding signal which gets emitted to move the handle when the user is using key bindings to move it.

Parameters

widget	the object that received the signal
scroll_type	a GtkScrollType
user_data	user data set when the signal handler was connected.

Flags: Action

Since: 2.0

The "toggle-handle-focus" signal

```
gboolean
user_function (GtkPaned *widget,
               gpointer  user_data)
```

The ::toggle-handle-focus is a keybinding signal which gets emitted to accept the current position of the handle and then move focus to the next widget in the focus chain.

The default binding is Tab.

Parameters

widget	the object that received the signal
user_data	user data set when the signal handler was connected.

Flags: Action

Since: 2.0

GtkLayout

GtkLayout — Infinite scrollable area containing child widgets and/or custom drawing

Functions

GtkWidget *	gtk_layout_new ()
void	gtk_layout_put ()
void	gtk_layout_move ()
void	gtk_layout_set_size ()
void	gtk_layout_get_size ()
GtkAdjustment *	gtk_layout_get_hadjustment ()
GtkAdjustment *	gtk_layout_get_vadjustment ()
void	gtk_layout_set_hadjustment ()
void	gtk_layout_set_vadjustment ()

GdkWindow *	gtk_layout_get_bin_window ()

Properties

guint	height	Read / Write
guint	width	Read / Write

Child Properties

gint	x	Read / Write
gint	y	Read / Write

Types and Values

struct	GtkLayout

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkContainer
            └── GtkLayout
```

Implemented Interfaces

GtkLayout implements AtkImplementorIface, GtkBuildable and GtkScrollable.

Includes

```
#include <gtk/gtk.h>
```

Description

GtkLayout is similar to GtkDrawingArea in that it's a "blank slate" and doesn't do anything except paint a blank background by default. It's different in that it supports scrolling natively due to implementing GtkScrollable, and can contain child widgets since it's a GtkContainer.

If you just want to draw, a GtkDrawingArea is a better choice since it has lower overhead. If you just need to position child widgets at specific points, then GtkFixed provides that functionality on its own.

When handling expose events on a GtkLayout, you must draw to the GdkWindow returned by `gtk_layout_get_bin_window()`, rather than to the one returned by `gtk_widget_get_window()` as you would for a GtkDrawingArea.

Functions

gtk_layout_new ()

```
GtkWidget *
gtk_layout_new (GtkAdjustment *hadjustment,
                GtkAdjustment *vadjustment);
```

Creates a new GtkLayout. Unless you have a specific adjustment you'd like the layout to use for scrolling, pass `NULL` for `hadjustment` and `vadjustment` .

Parameters

hadjustment	horizontal scroll adjustment, or `NULL`.	[allow-none]
vadjustment	vertical scroll adjustment, or `NULL`.	[allow-none]

Returns

a new GtkLayout

---

gtk_layout_put ()

```
void
gtk_layout_put (GtkLayout *layout,
                GtkWidget *child_widget,
                gint x,
                gint y);
```

Adds `child_widget` to `layout` , at position (`x` ,`y`). `layout` becomes the new parent container of `child_widget` .

Parameters

layout	a GtkLayout
child_widget	child widget
x	X position of child widget
y	Y position of child widget

---

gtk_layout_move ()

```
void
gtk_layout_move (GtkLayout *layout,
                 GtkWidget *child_widget,
                 gint x,
                 gint y);
```

Moves a current child of `layout` to a new position.

Parameters

layout	a GtkLayout

child_widget	a current child of `layout`
x	X position to move to
y	Y position to move to

---

gtk_layout_set_size ()

```
void
gtk_layout_set_size (GtkLayout *layout,
                     guint width,
                     guint height);
```

Sets the size of the scrollable area of the layout.

Parameters

layout	a GtkLayout
width	width of entire scrollable area
height	height of entire scrollable area

---

gtk_layout_get_size ()

```
void
gtk_layout_get_size (GtkLayout *layout,
                     guint *width,
                     guint *height);
```

Gets the size that has been set on the layout, and that determines the total extents of the layout's scrollbar area. See `gtk_layout_set_size()`.

Parameters

layout	a GtkLayout	
width	location to store the width set on `layout` , or `NULL`.	[out][allow-none]
height	location to store the height set on `layout` , or `NULL`.	[out][allow-none]

---

gtk_layout_get_hadjustment ()

```
GtkAdjustment *
gtk_layout_get_hadjustment (GtkLayout *layout);
```

`gtk_layout_get_hadjustment` has been deprecated since version 3.0 and should not be used in newly-written code.

Use `gtk_scrollable_get_hadjustment()`

This function should only be called after the layout has been placed in a GtkScrolledWindow or otherwise configured for scrolling. It returns the GtkAdjustment used for communication between the horizontal scrollbar

and `layout` .

See GtkScrolledWindow, GtkScrollbar, GtkAdjustment for details.

Parameters

layout	a GtkLayout

Returns

horizontal scroll adjustment.

[transfer none]

---

gtk_layout_get_vadjustment ()

```
GtkAdjustment *
gtk_layout_get_vadjustment (GtkLayout *layout);
```

`gtk_layout_get_vadjustment` has been deprecated since version 3.0 and should not be used in newly-written code.

Use `gtk_scrollable_get_vadjustment()`

This function should only be called after the layout has been placed in a GtkScrolledWindow or otherwise configured for scrolling. It returns the GtkAdjustment used for communication between the vertical scrollbar and `layout` .

See GtkScrolledWindow, GtkScrollbar, GtkAdjustment for details.

Parameters

layout	a GtkLayout

Returns

vertical scroll adjustment.

[transfer none]

---

gtk_layout_set_hadjustment ()

```
void
gtk_layout_set_hadjustment (GtkLayout *layout,
                            GtkAdjustment *adjustment);
```

`gtk_layout_set_hadjustment` has been deprecated since version 3.0 and should not be used in newly-written code.

Use `gtk_scrollable_set_hadjustment()`

Sets the horizontal scroll adjustment for the layout.

See GtkScrolledWindow, GtkScrollbar, GtkAdjustment for details.

Parameters

layout	a GtkLayout	
adjustment	new scroll adjustment.	[allow-none]

---

gtk_layout_set_vadjustment ()

```
void
gtk_layout_set_vadjustment (GtkLayout *layout,
                            GtkAdjustment *adjustment);
```

`gtk_layout_set_vadjustment` has been deprecated since version 3.0 and should not be used in newly-written code.

Use `gtk_scrollable_set_vadjustment()`

Sets the vertical scroll adjustment for the layout.

See GtkScrolledWindow, GtkScrollbar, GtkAdjustment for details.

Parameters

layout	a GtkLayout	
adjustment	new scroll adjustment.	[allow-none]

---

gtk_layout_get_bin_window ()

```
GdkWindow *
gtk_layout_get_bin_window (GtkLayout *layout);
```

Retrieve the bin window of the layout used for drawing operations.

Parameters

layout	a GtkLayout

Returns

a GdkWindow.

[transfer none]

Since: 2.14

Types and Values

struct GtkLayout

```
struct GtkLayout;
```

Property Details

The “height” property

  “height” guint

The height of the layout.

Flags: Read / Write

Allowed values: <= G_MAXINT

Default value: 100

The “width” property

  “width” guint

The width of the layout.

Flags: Read / Write

Allowed values: <= G_MAXINT

Default value: 100

Child Property Details

The “x” child property

  “x” gint

X position of child widget.

Flags: Read / Write

Default value: 0

The “y” child property

  “y” gint

Y position of child widget.

Flags: Read / Write

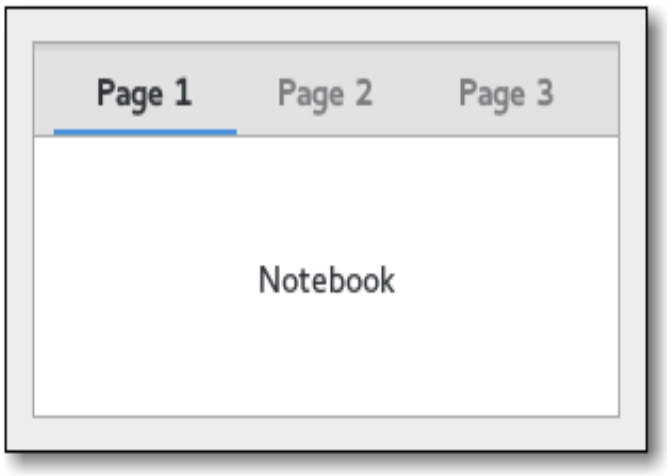
Default value: 0

See Also

GtkDrawingArea, GtkFixed

GtkNotebook

GtkNotebook — A tabbed notebook container



Functions

GtkWidget *	gtk_notebook_new ()
gint	gtk_notebook_append_page ()
gint	gtk_notebook_append_page_menu ()
gint	gtk_notebook_prepend_page ()
gint	gtk_notebook_prepend_page_menu ()
gint	gtk_notebook_insert_page ()
gint	gtk_notebook_insert_page_menu ()
void	gtk_notebook_remove_page ()
void	gtk_notebook_detach_tab ()
gint	gtk_notebook_page_num ()
void	gtk_notebook_next_page ()
void	gtk_notebook_prev_page ()
void	gtk_notebook_reorder_child ()
void	gtk_notebook_set_tab_pos ()
void	gtk_notebook_set_show_tabs ()
void	gtk_notebook_set_show_border ()
void	gtk_notebook_set_scrollable ()
void	gtk_notebook_popup_enable ()
void	gtk_notebook_popup_disable ()
gint	gtk_notebook_get_current_page ()
GtkWidget *	gtk_notebook_get_menu_label ()
GtkWidget *	gtk_notebook_get_nth_page ()
gint	gtk_notebook_get_n_pages ()
GtkWidget *	gtk_notebook_get_tab_label ()

void	gtk_notebook_set_menu_label ()
void	gtk_notebook_set_menu_label_text ()
void	gtk_notebook_set_tab_label ()
void	gtk_notebook_set_tab_label_text ()
void	gtk_notebook_set_tab_reorderable ()
void	gtk_notebook_set_tab_detachable ()
const gchar *	gtk_notebook_get_menu_label_text ()
gboolean	gtk_notebook_get_scrollable ()
gboolean	gtk_notebook_get_show_border ()
gboolean	gtk_notebook_get_show_tabs ()
const gchar *	gtk_notebook_get_tab_label_text ()
GtkPositionType	gtk_notebook_get_tab_pos ()
gboolean	gtk_notebook_get_tab_reorderable ()
gboolean	gtk_notebook_get_tab_detachable ()
guint16	gtk_notebook_get_tab_hborder ()
guint16	gtk_notebook_get_tab_vborder ()
void	gtk_notebook_set_current_page ()
void	gtk_notebook_set_group_name ()
const gchar *	gtk_notebook_get_group_name ()
void	gtk_notebook_set_action_widget ()
GtkWidget *	gtk_notebook_get_action_widget ()

Properties

gboolean	enable-popup	Read / Write
gchar *	group-name	Read / Write
gint	page	Read / Write
gboolean	scrollable	Read / Write
gboolean	show-border	Read / Write
gboolean	show-tabs	Read / Write
GtkPositionType	tab-pos	Read / Write

Child Properties

gboolean	detachable	Read / Write
gchar *	menu-label	Read / Write
gint	position	Read / Write
gboolean	reorderable	Read / Write
gboolean	tab-expand	Read / Write
gboolean	tab-fill	Read / Write
gchar *	tab-label	Read / Write

Style Properties

gint	arrow-spacing	Read
gboolean	has-backward-stepper	Read
gboolean	has-forward-stepper	Read
gboolean	has-secondary-backward-stepper	Read
gboolean	has-secondary-forward-stepper	Read
gboolean	has-tab-gap	Read

gint	initial-gap	Read
gint	tab-curvature	Read
gint	tab-overlap	Read

Signals

gboolean	change-current-page	Action
GtkNotebook*	create-window	Run Last
gboolean	focus-tab	Action
void	move-focus-out	Action
void	page-added	Run Last
void	page-removed	Run Last
void	page-reordered	Run Last
gboolean	reorder-tab	Action
gboolean	select-page	Action
void	switch-page	Run Last

Types and Values

struct	GtkNotebook

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkContainer
            └── GtkNotebook
```

Implemented Interfaces

GtkNotebook implements AtkImplementorIface and GtkBuildable.

Includes

```
#include <gtk/gtk.h>
```

Description

The GtkNotebook widget is a GtkContainer whose children are pages that can be switched between using tab labels along one edge.

There are many configuration options for GtkNotebook. Among other things, you can choose on which edge the tabs appear (see `gtk_notebook_set_tab_pos()`), whether, if there are too many tabs to fit the notebook should be made bigger or scrolling arrows added (see `gtk_notebook_set_scrollable()`), and whether there will be a popup menu allowing the users to switch pages. (see `gtk_notebook_popup_enable()`, `gtk_notebook_popup_disable()`)

GtkNotebook as GtkBuildable

The GtkNotebook implementation of the GtkBuildable interface supports placing children into tabs by specifying “tab” as the “type” attribute of a <child> element. Note that the content of the tab must be created before the tab can be filled. A tab child can be specified without specifying a <child> type attribute.

To add a child widget in the notebooks action area, specify "action-start" or “action-end” as the “type” attribute of the <child> element.

An example of a UI definition fragment with GtkNotebook:

```
<object class="GtkNotebook">
  <child>
    <object class="GtkLabel" id="notebook-content">
      <property name="label">Content</property>
    </object>
  </child>
  <child type="tab">
    <object class="GtkLabel" id="notebook-tab">
      <property name="label">Tab</property>
    </object>
  </child>
</object>
```

CSS nodes

```
notebook
├── header.top
│   ├── [<action widget>]
│   ├── tabs
│   │   ├── [arrow]
│   │   ├── tab
│   │   │   └── <tab label>
┊   ┊   ┊   ┊
│   │   │   ├── tab[.reorderable-page]
│   │   │   │   └── <tab label>
│   │   │   └── [arrow]
│   │   └── [<action widget>]
│
└── stack
    ├── <child>
    ┊
    └── <child>
```

GtkNotebook has a main CSS node with name notebook, a subnode with name header and below that a subnode with name tabs which contains one subnode per tab with name tab.

If action widgets are present, their CSS nodes are placed next to the tabs node. If the notebook is scrollable, CSS nodes with name arrow are placed as first and last child of the tabs node.

The main node gets the .frame style class when the notebook has a border (see `gtk_notebook_set_show_border()`).

The header node gets one of the style class .top, .bottom, .left or .right, depending on where the tabs are placed. For reorderable pages, the tab node gets the .reorderable-page class.

A tab node gets the .dnd style class while it is moved with drag-and-drop.

The nodes are always arranged from left-to-right, regarldess of text direction.

Functions

gtk_notebook_new ()

```
GtkWidget *
gtk_notebook_new (void);
```

Creates a new GtkNotebook widget with no pages.

Returns

the newly created GtkNotebook

---

gtk_notebook_append_page ()

```
gint
gtk_notebook_append_page (GtkNotebook *notebook,
                          GtkWidget *child,
                          GtkWidget *tab_label);
```

Appends a page to `notebook` .

Parameters

notebook	a GtkNotebook	
child	the GtkWidget to use as the contents of the page	
tab_label	the GtkWidget to be used as the label for the page, or `NULL` to use the default label, “page N”.	[allow-none]

Returns

the index (starting from 0) of the appended page in the notebook, or -1 if function fails

---

gtk_notebook_append_page_menu ()

```
gint
gtk_notebook_append_page_menu (GtkNotebook *notebook,
                               GtkWidget *child,
                               GtkWidget *tab_label,
                               GtkWidget *menu_label);
```

Appends a page to `notebook` , specifying the widget to use as the label in the popup menu.

Parameters

notebook	a GtkNotebook	
child	the GtkWidget to use as the contents of the page	
tab_label	the GtkWidget to be used as the label for the page, or `NULL` to use the default label, “page N”.	[allow-none]
menu_label	the widget to use as a label for the page-switch menu, if that is enabled. If `NULL`, and `tab_label` is a GtkLabel or `NULL`, then the menu label will be a newly created label with the same text as `tab_label` ; if `tab_label` is not a GtkLabel, `menu_label` must be specified if the page-switch menu is to be used.	[allow-none]

Returns

the index (starting from 0) of the appended page in the notebook, or -1 if function fails

---

gtk_notebook_prepend_page ()

```
gint
gtk_notebook_prepend_page (GtkNotebook *notebook,
                           GtkWidget *child,
                           GtkWidget *tab_label);
```

Prepends a page to `notebook` .

Parameters

notebook	a GtkNotebook	
child	the GtkWidget to use as the contents of the page	
tab_label	the GtkWidget to be used as the label for the page, or `NULL` to use the default label, “page N”.	[allow-none]

Returns

the index (starting from 0) of the prepended page in the notebook, or -1 if function fails

---

gtk_notebook_prepend_page_menu ()

```
gint
gtk_notebook_prepend_page_menu (GtkNotebook *notebook,
                                GtkWidget *child,
```

```
                                GtkWidget *tab_label,
                                GtkWidget *menu_label);
```

Prepends a page to `notebook` , specifying the widget to use as the label in the popup menu.

Parameters

notebook	a GtkNotebook	
child	the GtkWidget to use as the contents of the page	
tab_label	the GtkWidget to be used as the label for the page, or `NULL` to use the default label, “page N”.	[allow-none]
menu_label	the widget to use as a label for the page-switch menu, if that is enabled. If `NULL`, and `tab_label` is a GtkLabel or `NULL`, then the menu label will be a newly created label with the same text as `tab_label` ; if `tab_label` is not a GtkLabel, `menu_label` must be specified if the page-switch menu is to be used.	[allow-none]

Returns

the index (starting from 0) of the prepended page in the notebook, or -1 if function fails

---

gtk_notebook_insert_page ()

```
gint
gtk_notebook_insert_page (GtkNotebook *notebook,
                          GtkWidget *child,
                          GtkWidget *tab_label,
                          gint position);
```

Insert a page into `notebook` at the given position.

Parameters

notebook	a GtkNotebook	
child	the GtkWidget to use as the contents of the page	
tab_label	the GtkWidget to be used as the label for the page, or `NULL` to use the default label, “page N”.	[allow-none]
position	the index (starting at 0) at which to insert the page, or -1 to append the page after all other pages	

Returns

the index (starting from 0) of the inserted page in the notebook, or -1 if function fails

---

gtk_notebook_insert_page_menu ()

```
gint
gtk_notebook_insert_page_menu (GtkNotebook *notebook,
                               GtkWidget *child,
                               GtkWidget *tab_label,
                               GtkWidget *menu_label,
                               gint position);
```

Insert a page into `notebook` at the given position, specifying the widget to use as the label in the popup menu.

Parameters

notebook	a GtkNotebook	
child	the GtkWidget to use as the contents of the page	
tab_label	the GtkWidget to be used as the label for the page, or `NULL` to use the default label, “page N”.	[allow-none]
menu_label	the widget to use as a label for the page-switch menu, if that is enabled. If `NULL`, and `tab_label` is a GtkLabel or `NULL`, then the menu label will be a newly created label with the same text as `tab_label` ; if `tab_label` is not a GtkLabel, `menu_label` must be specified if the page-switch menu is to be used.	[allow-none]
position	the index (starting at 0) at which to insert the page, or -1 to append the page after all other pages.	

Returns

the index (starting from 0) of the inserted page in the notebook

---

gtk_notebook_remove_page ()

```
void
gtk_notebook_remove_page (GtkNotebook *notebook,
                          gint page_num);
```

Removes a page from the notebook given its index in the notebook.

Parameters

notebook	a GtkNotebook
page_num	the index of a notebook page, starting from 0. If -1, the last page will be removed.

---

gtk_notebook_detach_tab ()

```
void
gtk_notebook_detach_tab (GtkNotebook *notebook,
                         GtkWidget *child);
```

Removes the child from the notebook.

This function is very similar to `gtk_container_remove()`, but additionally informs the notebook that the removal is happening as part of a tab DND operation, which should not be cancelled.

Parameters

notebook	a GtkNotebook
child	a child

Since: 3.16

---

gtk_notebook_page_num ()

```
gint
gtk_notebook_page_num (GtkNotebook *notebook,
                       GtkWidget *child);
```

Finds the index of the page which contains the given child widget.

Parameters

notebook	a GtkNotebook
child	a GtkWidget

Returns

the index of the page containing `child` , or -1 if `child` is not in the notebook

---

gtk_notebook_next_page ()

```
void
gtk_notebook_next_page (GtkNotebook *notebook);
```

Switches to the next page. Nothing happens if the current page is the last page.

Parameters

notebook	a GtkNotebook

---

gtk_notebook_prev_page ()

```
void
gtk_notebook_prev_page (GtkNotebook *notebook);
```

Switches to the previous page. Nothing happens if the current page is the first page.

Parameters

notebook	a GtkNotebook

---

gtk_notebook_reorder_child ()

```
void
gtk_notebook_reorder_child (GtkNotebook *notebook,
                            GtkWidget *child,
                            gint position);
```

Reorders the page containing `child` , so that it appears in position `position` . If `position` is greater than or equal to the number of children in the list or negative, `child` will be moved to the end of the list.

Parameters

notebook	a GtkNotebook
child	the child to move
position	the new position, or -1 to move to the end

---

gtk_notebook_set_tab_pos ()

```
void
gtk_notebook_set_tab_pos (GtkNotebook *notebook,
                          GtkPositionType pos);
```

Sets the edge at which the tabs for switching pages in the notebook are drawn.

Parameters

notebook	a GtkNotebook.
pos	the edge to draw the tabs at

---

gtk_notebook_set_show_tabs ()

```
void
gtk_notebook_set_show_tabs (GtkNotebook *notebook,
                            gboolean show_tabs);
```

Sets whether to show the tabs for the notebook or not.

Parameters

notebook	a GtkNotebook
show_tabs	TRUE if the tabs should be shown

---

gtk_notebook_set_show_border ()

```
void
gtk_notebook_set_show_border (GtkNotebook *notebook,
                              gboolean show_border);
```

Sets whether a bevel will be drawn around the notebook pages. This only has a visual effect when the tabs are not shown. See gtk_notebook_set_show_tabs().

Parameters

notebook	a GtkNotebook
show_border	TRUE if a bevel should be drawn around the notebook

---

gtk_notebook_set_scrollable ()

```
void
gtk_notebook_set_scrollable (GtkNotebook *notebook,
                             gboolean scrollable);
```

Sets whether the tab label area will have arrows for scrolling if there are too many tabs to fit in the area.

Parameters

notebook	a GtkNotebook
scrollable	TRUE if scroll arrows should be added

---

gtk_notebook_popup_enable ()

```
void
gtk_notebook_popup_enable (GtkNotebook *notebook);
```

Enables the popup menu: if the user clicks with the right mouse button on the tab labels, a menu with all the pages will be popped up.

Parameters

notebook	a GtkNotebook

---

gtk_notebook_popup_disable ()

```
void
gtk_notebook_popup_disable (GtkNotebook *notebook);
```

Disables the popup menu.

Parameters

notebook	a GtkNotebook

---

gtk_notebook_get_current_page ()

```
gint
gtk_notebook_get_current_page (GtkNotebook *notebook);
```

Returns the page number of the current page.

Parameters

notebook	a GtkNotebook

Returns

the index (starting from 0) of the current page in the notebook. If the notebook has no pages, then -1 will be returned.

---

gtk_notebook_get_menu_label ()

```
GtkWidget *
gtk_notebook_get_menu_label (GtkNotebook *notebook,
                             GtkWidget *child);
```

Retrieves the menu label widget of the page containing `child`.

Parameters

notebook	a GtkNotebook
child	a widget contained in a page of `notebook`

Returns

the menu label, or `NULL` if the notebook page does not have a menu label other than the default (the tab label).

[nullable][transfer none]

---

gtk_notebook_get_nth_page ()

```
GtkWidget *
gtk_notebook_get_nth_page (GtkNotebook *notebook,
                           gint page_num);
```

Returns the child widget contained in page number `page_num` .

Parameters

notebook	a GtkNotebook
page_num	the index of a page in the notebook, or -1 to get the last page

Returns

the child widget, or `NULL` if `page_num` is out of bounds.

[nullable][transfer none]

---

gtk_notebook_get_n_pages ()

```
gint
gtk_notebook_get_n_pages (GtkNotebook *notebook);
```

Gets the number of pages in a notebook.

Parameters

notebook	a GtkNotebook

Returns

the number of pages in the notebook

Since: 2.2

---

gtk_notebook_get_tab_label ()

```
GtkWidget *
gtk_notebook_get_tab_label (GtkNotebook *notebook,
                            GtkWidget *child);
```

Returns the tab label widget for the page `child` . `NULL` is returned if `child` is not in `notebook` or if no tab label has specifically been set for `child` .

Parameters

notebook	a GtkNotebook
child	the page

Returns

the tab label.

[transfer none][nullable]

---

gtk_notebook_set_menu_label ()

```
void
gtk_notebook_set_menu_label (GtkNotebook *notebook,
                             GtkWidget *child,
                             GtkWidget *menu_label);
```

Changes the menu label for the page containing `child` .

Parameters

notebook	a GtkNotebook
child	the child widget
menu_label	the menu label, or `NULL` for default. [allow-none]

---

gtk_notebook_set_menu_label_text ()

```
void
gtk_notebook_set_menu_label_text (GtkNotebook *notebook,
                                  GtkWidget *child,
                                  const gchar *menu_text);
```

Creates a new label and sets it as the menu label of `child` .

Parameters

notebook	a GtkNotebook
child	the child widget
menu_text	the label text

---

gtk_notebook_set_tab_label ()

```
void
gtk_notebook_set_tab_label (GtkNotebook *notebook,
                            GtkWidget *child,
                            GtkWidget *tab_label);
```

Changes the tab label for `child` . If `NULL` is specified for `tab_label` , then the page will have the label “page N”.

Parameters

notebook	a GtkNotebook
child	the page
tab_label	the tab label widget to use, or `NULL` [allow-none] for default tab label.

---

gtk_notebook_set_tab_label_text ()

```
void
gtk_notebook_set_tab_label_text (GtkNotebook *notebook,
                                 GtkWidget *child,
                                 const gchar *tab_text);
```

Creates a new label and sets it as the tab label for the page containing `child` .

Parameters

notebook	a GtkNotebook
child	the page
tab_text	the label text

---

gtk_notebook_set_tab_reorderable ()

```
void
gtk_notebook_set_tab_reorderable (GtkNotebook *notebook,
                                  GtkWidget *child,
                                  gboolean reorderable);
```

Sets whether the notebook tab can be reordered via drag and drop or not.

Parameters

notebook	a GtkNotebook
child	a child GtkWidget
reorderable	whether the tab is reorderable or not

Since: 2.10

---

gtk_notebook_set_tab_detachable ()

```
void
gtk_notebook_set_tab_detachable (GtkNotebook *notebook,
                                 GtkWidget *child,
                                 gboolean detachable);
```

Sets whether the tab can be detached from `notebook` to another notebook or widget.

Note that 2 notebooks must share a common group identificator (see `gtk_notebook_set_group_name()`) to allow automatic tabs interchange between them.

If you want a widget to interact with a notebook through DnD (i.e.: accept dragged tabs from it) it must be set as a drop destination and accept the target “GTK_NOTEBOOK_TAB”. The notebook will fill the selection with a GtkWidget** pointing to the child widget that corresponds to the dropped tab.

Note that you should use `gtk_notebook_detach_tab()` instead of `gtk_container_remove()` if you want to remove the tab from the source notebook as part of accepting a drop. Otherwise, the source notebook will think that the dragged tab was removed from underneath the ongoing drag operation, and will initiate a drag cancel animation.

```
1   static void
2   on_drag_data_received (GtkWidget        *widget,
3                          GdkDragContext   *context,
4                          gint              x,
5                          gint              y,
6                          GtkSelectionData *data,
7                          guint             info,
8                          guint             time,
9                          gpointer          user_data)
10  {
11    GtkWidget *notebook;
12    GtkWidget **child;
13
14    notebook = gtk_drag_get_source_widget (context);
15    child = (void*) gtk_selection_data_get_data (data);
16
17    // process_widget (*child);
18
19    gtk_notebook_detach_tab (GTK_NOTEBOOK (notebook), *child);
20  }
```

If you want a notebook to accept drags from other widgets, you will have to set your own DnD code to do it.

Parameters

notebook	a GtkNotebook
child	a child GtkWidget
detachable	whether the tab is detachable or not

Since: 2.10

---

gtk_notebook_get_menu_label_text ()

```
const gchar *
gtk_notebook_get_menu_label_text (GtkNotebook *notebook,
                                  GtkWidget *child);
```

Retrieves the text of the menu label for the page containing `child` .

Parameters

notebook	a GtkNotebook
child	the child widget of a page of the notebook.

Returns

the text of the tab label, or `NULL` if the widget does not have a menu label other than the default menu label, or the menu label widget is not a GtkLabel. The string is owned by the widget and must not be freed.

[nullable]

---

gtk_notebook_get_scrollable ()

```
gboolean
gtk_notebook_get_scrollable (GtkNotebook *notebook);
```

Returns whether the tab label area has arrows for scrolling. See `gtk_notebook_set_scrollable()`.

Parameters

notebook	a GtkNotebook

Returns

`TRUE` if arrows for scrolling are present

---

gtk_notebook_get_show_border ()

```
gboolean
gtk_notebook_get_show_border (GtkNotebook *notebook);
```

Returns whether a bevel will be drawn around the notebook pages. See `gtk_notebook_set_show_border()`.

Parameters

notebook	a GtkNotebook

Returns

`TRUE` if the bevel is drawn

---

gtk_notebook_get_show_tabs ()

```
gboolean
gtk_notebook_get_show_tabs (GtkNotebook *notebook);
```

Returns whether the tabs of the notebook are shown. See `gtk_notebook_set_show_tabs()`.

Parameters

notebook	a GtkNotebook

Returns

`TRUE` if the tabs are shown

---

gtk_notebook_get_tab_label_text ()

```
const gchar *
gtk_notebook_get_tab_label_text (GtkNotebook *notebook,
                                 GtkWidget *child);
```

Retrieves the text of the tab label for the page containing `child`.

Parameters

notebook	a GtkNotebook
child	a widget contained in a page of `notebook`

Returns

the text of the tab label, or `NULL` if the tab label widget is not a GtkLabel. The string is owned by the widget and must not be freed.

[nullable]

---

gtk_notebook_get_tab_pos ()

```
GtkPositionType
gtk_notebook_get_tab_pos (GtkNotebook *notebook);
```

Gets the edge at which the tabs for switching pages in the notebook are drawn.

Parameters

notebook	a GtkNotebook

Returns

the edge at which the tabs are drawn

---

gtk_notebook_get_tab_reorderable ()

```
gboolean
gtk_notebook_get_tab_reorderable (GtkNotebook *notebook,
                                  GtkWidget *child);
```

Gets whether the tab can be reordered via drag and drop or not.

Parameters

notebook	a GtkNotebook
child	a child GtkWidget

Returns

`TRUE` if the tab is reorderable.

Since: 2.10

---

gtk_notebook_get_tab_detachable ()

```
gboolean
gtk_notebook_get_tab_detachable (GtkNotebook *notebook,
                                 GtkWidget *child);
```

Returns whether the tab contents can be detached from `notebook` .

Parameters

notebook	a GtkNotebook
child	a child GtkWidget

Returns

`TRUE` if the tab is detachable.

Since: 2.10

---

gtk_notebook_get_tab_hborder ()

```
guint16
gtk_notebook_get_tab_hborder (GtkNotebook *notebook);
```

`gtk_notebook_get_tab_hborder` has been deprecated since version 3.4 and should not be used in newly-written code.

this function returns zero

Returns the horizontal width of a tab border.

Parameters

notebook	a GtkNotebook

Returns

horizontal width of a tab border

Since: 2.22

---

gtk_notebook_get_tab_vborder ()

```
guint16
gtk_notebook_get_tab_vborder (GtkNotebook *notebook);
```

`gtk_notebook_get_tab_vborder` has been deprecated since version 3.4 and should not be used in newly-written code.

this function returns zero

Returns the vertical width of a tab border.

Parameters

notebook	a GtkNotebook

Returns

vertical width of a tab border

Since: 2.22

---

gtk_notebook_set_current_page ()

```
void
gtk_notebook_set_current_page (GtkNotebook *notebook,
                               gint page_num);
```

Switches to the page number `page_num` .

Note that due to historical reasons, GtkNotebook refuses to switch to a page unless the child widget is visible. Therefore, it is recommended to show child widgets before adding them to a notebook.

Parameters

notebook	a GtkNotebook
page_num	index of the page to switch to, starting from 0. If negative, the last page will be used. If greater than the number of pages in the notebook,

nothing will be done.

---

gtk_notebook_set_group_name ()

```
void
gtk_notebook_set_group_name (GtkNotebook *notebook,
                             const gchar *group_name);
```

Sets a group name for `notebook` .

Notebooks with the same name will be able to exchange tabs via drag and drop. A notebook with a `NULL` group name will not be able to exchange tabs with any other notebook.

Parameters

notebook	a GtkNotebook	
group_name	the name of the notebook group, or `NULL` to unset it.	[allow-none]

Since: 2.24

---

gtk_notebook_get_group_name ()

```
const gchar *
gtk_notebook_get_group_name (GtkNotebook *notebook);
```

Gets the current group name for `notebook` .

Parameters

notebook	a GtkNotebook

Returns

the group name, or `NULL` if none is set.

[nullable][transfer none]

Since: 2.24

---

gtk_notebook_set_action_widget ()

```
void
gtk_notebook_set_action_widget (GtkNotebook *notebook,
                                GtkWidget *widget,
                                GtkPackType pack_type);
```

Sets `widget` as one of the action widgets. Depending on the pack type the widget will be placed before or after the tabs. You can use a GtkBox if you need to pack more than one widget on the same side.

Note that action widgets are “internal” children of the notebook and thus not included in the list returned from

gtk_container_foreach().

Parameters

notebook	a GtkNotebook
widget	a GtkWidget
pack_type	pack type of the action widget

Since: 2.20

---

gtk_notebook_get_action_widget ()

```
GtkWidget *
gtk_notebook_get_action_widget (GtkNotebook *notebook,
                                GtkPackType pack_type);
```

Gets one of the action widgets. See gtk_notebook_set_action_widget().

Parameters

notebook	a GtkNotebook
pack_type	pack type of the action widget to receive

Returns

The action widget with the given `pack_type` or `NULL` when this action widget has not been set.

[nullable][transfer none]

Since: 2.20

Types and Values

struct GtkNotebook

```
struct GtkNotebook;
```

Property Details

The “enable-popup” property

```
  “enable-popup”             gboolean
```

If TRUE, pressing the right mouse button on the notebook pops up a menu that you can use to go to a page.

Flags: Read / Write

Default value: FALSE

The “group-name” property

“group-name” gchar *

Group name for tab drag and drop.

Flags: Read / Write

Default value: NULL

Since: 2.24

The “page” property

“page” gint

The index of the current page.

Flags: Read / Write

Allowed values: >= -1

Default value: -1

The “scrollable” property

“scrollable” gboolean

If TRUE, scroll arrows are added if there are too many tabs to fit.

Flags: Read / Write

Default value: FALSE

The “show-border” property

“show-border” gboolean

Whether the border should be shown.

Flags: Read / Write

Default value: TRUE

The “show-tabs” property

“show-tabs” gboolean

Whether tabs should be shown.

Flags: Read / Write

Default value: TRUE

The “tab-pos” property

“tab-pos” GtkPositionType

Which side of the notebook holds the tabs.

Flags: Read / Write

Default value: GTK_POS_TOP

Child Property Details

The “detachable” child property

“detachable” gboolean

Whether the tab is detachable.

Flags: Read / Write

Default value: FALSE

The “menu-label” child property

“menu-label” gchar *

The string displayed in the child's menu entry.

Flags: Read / Write

Default value: NULL

The “position” child property

“position” gint

The index of the child in the parent.

Flags: Read / Write

Allowed values: >= -1

Default value: 0

The “reorderable” child property

“reorderable” gboolean

Whether the tab is reorderable by user action.

Flags: Read / Write

Default value: FALSE

The “tab-expand” child property

“tab-expand” gboolean

Whether to expand the child's tab.

Flags: Read / Write

Default value: FALSE

The “tab-fill” child property

“tab-fill” gboolean

Whether the child's tab should fill the allocated area.

Flags: Read / Write

Default value: TRUE

The “tab-label” child property

“tab-label” gchar *

The string displayed on the child's tab label.

Flags: Read / Write

Default value: NULL

Style Property Details

The “arrow-spacing” style property

“arrow-spacing” gint

The "arrow-spacing" property defines the spacing between the scroll arrows and the tabs.

GtkNotebook:arrow-spacing has been deprecated since version 3.20 and should not be used in newly-written code.

This property is ignored. Use margins on arrows or the "tabs" node to achieve the same effect.

Flags: Read

Allowed values: >= 0

Default value: 0

Since: 2.10

The “has-backward-stepper” style property

“has-backward-stepper” gboolean

The “has-backward-stepper” property determines whether the standard backward arrow button is displayed.

Flags: Read

Default value: TRUE

Since: 2.4

The “has-forward-stepper” style property

“has-forward-stepper” gboolean

The “has-forward-stepper” property determines whether the standard forward arrow button is displayed.

Flags: Read

Default value: TRUE

Since: 2.4

The “has-secondary-backward-stepper” style property

“has-secondary-backward-stepper” gboolean

The “has-secondary-backward-stepper” property determines whether a second backward arrow button is displayed on the opposite end of the tab area.

Flags: Read

Default value: FALSE

Since: 2.4

The “has-secondary-forward-stepper” style property

“has-secondary-forward-stepper” gboolean

The “has-secondary-forward-stepper” property determines whether a second forward arrow button is displayed on the opposite end of the tab area.

Flags: Read

Default value: FALSE

Since: 2.4

The “has-tab-gap” style property

“has-tab-gap” gboolean

The "has-tab-gap" property defines whether the active tab is draw with a gap at the bottom. When TRUE the theme engine uses gtk_render_extension to draw the active tab. When FALSE gtk_render_background and gtk_render_frame are used.

GtkNotebook:has-tab-gap has been deprecated since version 3.20 and should not be used in newly-written code.

This function always behaves as if it was set to FALSE.

Flags: Read

Default value: TRUE

Since: 3.12

The “initial-gap” style property

“initial-gap” gint

The "initial-gap" property defines the minimum size for the initial gap between the first tab.

GtkNotebook:initial-gap has been deprecated since version 3.20 and should not be used in newly-written code.

The intial gap is ignored. Use margins on the header node to achieve the same effect.

Flags: Read

Allowed values: >= 0

Default value: 0

Since: 3.2

The “tab-curvature” style property

“tab-curvature” gint

The “tab-curvature” property defines size of tab curvature.

GtkNotebook:tab-curvature has been deprecated since version 3.20 and should not be used in newly-written code.

This property is ignored. Use margins on tab nodes to achieve the same effect.

Flags: Read

Allowed values: >= 0

Default value: 1

Since: 2.10

The “tab-overlap” style property

```
  “tab-overlap”              gint
```

The “tab-overlap” property defines size of tab overlap area.

`GtkNotebook:tab-overlap` has been deprecated since version 3.20 and should not be used in newly-written code.

This property is ignored. Use margins on tab nodes to achieve the same effect.

Flags: Read

Default value: 2

Since: 2.10

Signal Details

The “change-current-page” signal

```
gboolean
user_function (GtkNotebook *notebook,
               gint         arg1,
               gpointer     user_data)
```

Flags: Action

The “create-window” signal

```
GtkNotebook*
user_function (GtkNotebook *notebook,
               GtkWidget   *page,
               gint         x,
               gint         y,
               gpointer     user_data)
```

The ::create-window signal is emitted when a detachable tab is dropped on the root window.

A handler for this signal can create a window containing a notebook where the tab will be attached. It is also responsible for moving/resizing the window and adding the necessary properties to the notebook (e.g. the “group-name”).

Parameters

notebook	the GtkNotebook emitting the signal
page	the tab of `notebook` that is being

	detached
x	the X coordinate where the drop happens
y	the Y coordinate where the drop happens
user_data	user data set when the signal handler was connected.

Returns

a GtkNotebook that `page` should be added to, or `NULL`.

[transfer none]

Flags: Run Last

Since: 2.12

The `"focus-tab"` signal

```
gboolean
user_function (GtkNotebook   *notebook,
               GtkNotebookTab arg1,
               gpointer       user_data)
```

Flags: Action

The `"move-focus-out"` signal

```
void
user_function (GtkNotebook     *notebook,
               GtkDirectionType arg1,
               gpointer         user_data)
```

Flags: Action

The `"page-added"` signal

```
void
user_function (GtkNotebook *notebook,
               GtkWidget   *child,
               guint        page_num,
               gpointer     user_data)
```

the ::page-added signal is emitted in the notebook right after a page is added to the notebook.

Parameters

notebook	the GtkNotebook
child	the child GtkWidget affected
page_num	the new page number for `child`

user_data	user data set when the signal handler was connected.

Flags: Run Last

Since: 2.10

The “page-removed” signal

```
void
user_function (GtkNotebook *notebook,
               GtkWidget   *child,
               guint        page_num,
               gpointer     user_data)
```

the ::page-removed signal is emitted in the notebook right after a page is removed from the notebook.

Parameters

notebook	the GtkNotebook
child	the child GtkWidget affected
page_num	the `child` page number
user_data	user data set when the signal handler was connected.

Flags: Run Last

Since: 2.10

The “page-reordered” signal

```
void
user_function (GtkNotebook *notebook,
               GtkWidget   *child,
               guint        page_num,
               gpointer     user_data)
```

the ::page-reordered signal is emitted in the notebook right after a page has been reordered.

Parameters

notebook	the GtkNotebook
child	the child GtkWidget affected
page_num	the new page number for `child`
user_data	user data set when the signal handler was connected.

Flags: Run Last

Since: 2.10

The “reorder-tab” signal

```
gboolean
user_function (GtkNotebook     *notebook,
               GtkDirectionType arg1,
               gboolean         arg2,
               gpointer         user_data)
```

Flags: Action

The “select-page” signal

```
gboolean
user_function (GtkNotebook *notebook,
               gboolean     arg1,
               gpointer     user_data)
```

Flags: Action

The “switch-page” signal

```
void
user_function (GtkNotebook *notebook,
               GtkWidget   *page,
               guint        page_num,
               gpointer     user_data)
```

Emitted when the user or a function changes the current page.

Parameters

notebook	the object which received the signal.
page	the new current page
page_num	the index of the page
user_data	user data set when the signal handler was connected.

Flags: Run Last

See Also

GtkContainer

GtkExpander

GtkExpander — A container which can hide its child

Functions

GtkWidget *	gtk_expander_new ()
GtkWidget *	gtk_expander_new_with_mnemonic ()

void	gtk_expander_set_expanded ()
gboolean	gtk_expander_get_expanded ()
void	gtk_expander_set_spacing ()
gint	gtk_expander_get_spacing ()
void	gtk_expander_set_label ()
const gchar *	gtk_expander_get_label ()
void	gtk_expander_set_use_underline ()
gboolean	gtk_expander_get_use_underline ()
void	gtk_expander_set_use_markup ()
gboolean	gtk_expander_get_use_markup ()
void	gtk_expander_set_label_widget ()
GtkWidget *	gtk_expander_get_label_widget ()
void	gtk_expander_set_label_fill ()
gboolean	gtk_expander_get_label_fill ()
void	gtk_expander_set_resize_toplevel ()
gboolean	gtk_expander_get_resize_toplevel ()

Properties

gboolean	expanded	Read / Write / Construct
gchar *	label	Read / Write / Construct
gboolean	label-fill	Read / Write / Construct
GtkWidget *	label-widget	Read / Write
gboolean	resize-toplevel	Read / Write
gint	spacing	Read / Write
gboolean	use-markup	Read / Write / Construct
gboolean	use-underline	Read / Write / Construct

Style Properties

gint	expander-size	Read
gint	expander-spacing	Read

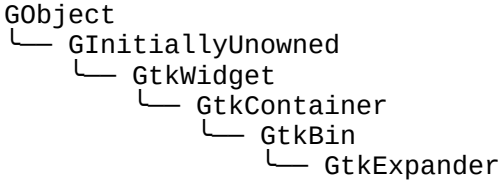
Signals

void	activate	Action

Types and Values

struct	GtkExpander
struct	GtkExpanderClass

Object Hierarchy



Implemented Interfaces

GtkExpander implements AtkImplementorIface and GtkBuildable.

Includes

```
#include <gtk/gtk.h>
```

Description

A GtkExpander allows the user to hide or show its child by clicking on an expander triangle similar to the triangles used in a GtkTreeView.

Normally you use an expander as you would use any other descendant of GtkBin; you create the child widget and use `gtk_container_add()` to add it to the expander. When the expander is toggled, it will take care of showing and hiding the child automatically.

Special Usage

There are situations in which you may prefer to show and hide the expanded widget yourself, such as when you want to actually create the widget at expansion time. In this case, create a GtkExpander but do not add a child to it. The expander widget has an "expanded" property which can be used to monitor its expansion state. You should watch this property with a signal connection as follows:

```
1    static void
2    expander_callback (GObject    *object,
3                       GParamSpec *param_spec,
4                       gpointer    user_data)
5    {
6      GtkExpander *expander;
7
8      expander = GTK_EXPANDER (object);
9
10     if (gtk_expander_get_expanded (expander))
11       {
12         // Show or create widgets
13       }
14     else
15       {
16         // Hide or destroy widgets
17       }
18   }
19
20   static void
21   create_expander (void)
22   {
23     GtkWidget *expander = gtk_expander_new_with_mnemonic ("_More Options");
24     g_signal_connect (expander, "notify::expanded",
25                       G_CALLBACK (expander_callback), NULL);
26
27     // ...
28   }
```

GtkExpander as GtkBuildable

The GtkExpander implementation of the GtkBuildable interface supports placing a child in the label position by specifying “label” as the “type” attribute of a <child> element. A normal content child can be specified without specifying a <child> type attribute.

An example of a UI definition fragment with GtkExpander:

```
<object class="GtkExpander">
  <child type="label">
    <object class="GtkLabel" id="expander-label"/>
  </child>
  <child>
    <object class="GtkEntry" id="expander-content"/>
  </child>
</object>
```

CSS nodes

```
expander
├── title
│   ├── arrow
│   └── <label widget>
└── <child>
```

GtkExpander has three CSS nodes, the main node with the name expander, a subnode with name title and node below it with name arrow. The arrow of an expander that is showing its child gets the :checked pseudoclass added to it.

Functions

gtk_expander_new ()

```
GtkWidget *
gtk_expander_new (const gchar *label);
```

Creates a new expander using `label` as the text of the label.

Parameters

label	the text of the label.	[nullable]

Returns

a new GtkExpander widget.

Since: 2.4

---

gtk_expander_new_with_mnemonic ()

```
GtkWidget *
gtk_expander_new_with_mnemonic (const gchar *label);
```

Creates a new expander using `label` as the text of the label. If characters in `label` are preceded by an underscore, they are underlined. If you need a literal underscore character in a label, use "__" (two underscores). The first underlined character represents a keyboard accelerator called a mnemonic. Pressing Alt and that key activates the button.

Parameters

label	the text of the label with an underscore in front of the mnemonic character.	[nullable]

Returns

a new GtkExpander widget.

Since: 2.4

---

gtk_expander_set_expanded ()

```
void
gtk_expander_set_expanded (GtkExpander *expander,
                           gboolean expanded);
```

Sets the state of the expander. Set to `TRUE`, if you want the child widget to be revealed, and `FALSE` if you want the child widget to be hidden.

Parameters

expander	a GtkExpander
expanded	whether the child widget is revealed

Since: 2.4

---

gtk_expander_get_expanded ()

```
gboolean
gtk_expander_get_expanded (GtkExpander *expander);
```

Queries a GtkExpander and returns its current state. Returns `TRUE` if the child widget is revealed.

See `gtk_expander_set_expanded()`.

Parameters

expander	a GtkExpander

Returns

the current state of the expander

Since: 2.4

---

gtk_expander_set_spacing ()

```
void
gtk_expander_set_spacing (GtkExpander *expander,
                          gint spacing);
```

`gtk_expander_set_spacing` has been deprecated since version 3.20 and should not be used in newly-written code.

Use margins on the child instead.

Sets the spacing field of `expander` , which is the number of pixels to place between expander and the child.

Parameters

expander	a GtkExpander
spacing	distance between the expander and child in pixels

Since: 2.4

---

gtk_expander_get_spacing ()

```
gint
gtk_expander_get_spacing (GtkExpander *expander);
```

`gtk_expander_get_spacing` has been deprecated since version 3.20 and should not be used in newly-written code.

Use margins on the child instead.

Gets the value set by `gtk_expander_set_spacing()`.

Parameters

expander	a GtkExpander

Returns

spacing between the expander and child

Since: 2.4

---

gtk_expander_set_label ()

```
void
gtk_expander_set_label (GtkExpander *expander,
                        const gchar *label);
```

Sets the text of the label of the expander to `label`.

This will also clear any previously set labels.

Parameters

expander	a GtkExpander	
label	a string.	[nullable]

Since: 2.4

---

gtk_expander_get_label ()

```
const gchar *
gtk_expander_get_label (GtkExpander *expander);
```

Fetches the text from a label widget including any embedded underlines indicating mnemonics and Pango markup, as set by `gtk_expander_set_label()`. If the label text has not been set the return value will be `NULL`. This will be the case if you create an empty button with `gtk_button_new()` to use as a container.

Note that this function behaved differently in versions prior to 2.14 and used to return the label text stripped of embedded underlines indicating mnemonics and Pango markup. This problem can be avoided by fetching the label text directly from the label widget.

Parameters

expander	a GtkExpander

Returns

The text of the label widget. This string is owned by the widget and must not be modified or freed.

[nullable]

Since: 2.4

---

gtk_expander_set_use_underline ()

```
void
gtk_expander_set_use_underline (GtkExpander *expander,
                                gboolean use_underline);
```

If true, an underline in the text of the expander label indicates the next character should be used for the mnemonic accelerator key.

Parameters

expander	a GtkExpander
use_underline	TRUE if underlines in the text indicate mnemonics

Since: 2.4

---

gtk_expander_get_use_underline ()

```
gboolean
gtk_expander_get_use_underline (GtkExpander *expander);
```

Returns whether an embedded underline in the expander label indicates a mnemonic. See gtk_expander_set_use_underline().

Parameters

expander	a GtkExpander

Returns

TRUE if an embedded underline in the expander label indicates the mnemonic accelerator keys

Since: 2.4

---

gtk_expander_set_use_markup ()

```
void
gtk_expander_set_use_markup (GtkExpander *expander,
                             gboolean use_markup);
```

Sets whether the text of the label contains markup in Pango's text markup language. See gtk_label_set_markup().

Parameters

expander	a GtkExpander
use_markup	TRUE if the label's text should be parsed for markup

Since: 2.4

---

gtk_expander_get_use_markup ()

```
gboolean
gtk_expander_get_use_markup (GtkExpander *expander);
```

Returns whether the label's text is interpreted as marked up with the Pango text markup language. See gtk_expander_set_use_markup().

Parameters

expander	a GtkExpander	

Returns

`TRUE` if the label's text will be parsed for markup

Since: 2.4

---

gtk_expander_set_label_widget ()

```
void
gtk_expander_set_label_widget (GtkExpander *expander,
                               GtkWidget *label_widget);
```

Set the label widget for the expander. This is the widget that will appear embedded alongside the expander arrow.

Parameters

expander	a GtkExpander	
label_widget	the new label widget.	[nullable]

Since: 2.4

---

gtk_expander_get_label_widget ()

```
GtkWidget *
gtk_expander_get_label_widget (GtkExpander *expander);
```

Retrieves the label widget for the frame. See `gtk_expander_set_label_widget()`.

Parameters

expander	a GtkExpander	

Returns

the label widget, or `NULL` if there is none.

[nullable][transfer none]

Since: 2.4

---

gtk_expander_set_label_fill ()

```
void
gtk_expander_set_label_fill (GtkExpander *expander,
                             gboolean label_fill);
```

Sets whether the label widget should fill all available horizontal space allocated to `expander` .

Parameters

expander	a GtkExpander
label_fill	`TRUE` if the label should should fill all available horizontal space

Since: 2.22

---

gtk_expander_get_label_fill ()

```
gboolean
gtk_expander_get_label_fill (GtkExpander *expander);
```

Returns whether the label widget will fill all available horizontal space allocated to `expander` .

Parameters

expander	a GtkExpander

Returns

`TRUE` if the label widget will fill all available horizontal space

Since: 2.22

---

gtk_expander_set_resize_toplevel ()

```
void
gtk_expander_set_resize_toplevel (GtkExpander *expander,
                                  gboolean resize_toplevel);
```

Sets whether the expander will resize the toplevel widget containing the expander upon resizing and collpasing.

Parameters

expander	a GtkExpander
resize_toplevel	whether to resize the toplevel

Since: 3.2

---

gtk_expander_get_resize_toplevel ()

```
gboolean
gtk_expander_get_resize_toplevel (GtkExpander *expander);
```

Returns whether the expander will resize the toplevel widget containing the expander upon resizing and collpasing.

Parameters

expander	a GtkExpander

Returns

the “resize toplevel” setting.

Since: 3.2

Types and Values

struct GtkExpander

```
struct GtkExpander;
```

---

struct GtkExpanderClass

```
struct GtkExpanderClass {
  GtkBinClass    parent_class;


  /* Key binding signal; to get notification on the expansion
   * state connect to notify:expanded.
   */
  void        (* activate) (GtkExpander *expander);
};
```

Members

`activate` ()	Keybinding signal is emitted when the user hits the Enter key.

Property Details

The “expanded” property

```
  “expanded”                 gboolean
```

Whether the expander has been opened to reveal the child widget.

Flags: Read / Write / Construct

Default value: FALSE

The “label” property

“label” gchar *

Text of the expander's label.

Flags: Read / Write / Construct

Default value: NULL

The “label-fill” property

“label-fill” gboolean

Whether the label widget should fill all available horizontal space.

Flags: Read / Write / Construct

Default value: FALSE

The “label-widget” property

“label-widget” GtkWidget *

A widget to display in place of the usual expander label.

Flags: Read / Write

The “resize-toplevel” property

“resize-toplevel” gboolean

When this property is TRUE, the expander will resize the toplevel widget containing the expander upon expanding and collapsing.

Flags: Read / Write

Default value: FALSE

Since: 3.2

The “spacing” property

“spacing” gint

Space to put between the label and the child when the expander is expanded.

GtkExpander:spacing has been deprecated since version 3.20 and should not be used in newly-written code.

This property is deprecated and ignored. Use margins on the child instead.

Flags: Read / Write

Allowed values: >= 0

Default value: 0

The “use-markup” property

“use-markup” gboolean

The text of the label includes XML markup. See pango_parse_markup().

Flags: Read / Write / Construct

Default value: FALSE

The “use-underline” property

“use-underline” gboolean

If set, an underline in the text indicates the next character should be used for the mnemonic accelerator key.

Flags: Read / Write / Construct

Default value: FALSE

Style Property Details

The “expander-size” style property

“expander-size” gint

The size of the expander arrow.

GtkExpander:expander-size has been deprecated since version 3.20 and should not be used in newly-written code.

Use CSS min-width and min-height instead.

Flags: Read

Allowed values: >= 0

Default value: 10

The “expander-spacing” style property

```
  “expander-spacing”         gint
```

Spaing around the expander arrow.

`GtkExpander:expander-spacing` has been deprecated since version 3.20 and should not be used in newly-written code.

Use CSS margins instead, the value of this style property is ignored.

Flags: Read

Allowed values: >= 0

Default value: 2

Signal Details

The “activate” signal

```
void
user_function (GtkExpander *expander,
               gpointer     user_data)
```

Flags: Action

GtkOrientable

GtkOrientable — An interface for flippable widgets

Functions

GtkOrientation	gtk_orientable_get_orientation ()
void	gtk_orientable_set_orientation ()

Properties

GtkOrientation	orientation	Read / Write

Types and Values

GtkOrientable

Object Hierarchy

```
GInterface
└── GtkOrientable
```

Prerequisites

GtkOrientable requires GObject.

Known Implementations

GtkOrientable is implemented by GtkAppChooserWidget, GtkBox, GtkButtonBox, GtkCellAreaBox, GtkCellRendererProgress, GtkCellView, GtkColorChooserWidget, GtkColorSelection, GtkFileChooserButton, GtkFileChooserWidget, GtkFlowBox, GtkFontChooserWidget, GtkFontSelection, GtkGrid, GtkHBox, GtkHButtonBox, GtkHPaned, GtkHScale, GtkHScrollbar, GtkHSeparator, GtkInfoBar, GtkLevelBar, GtkPaned, GtkProgressBar, GtkRange, GtkRecentChooserWidget, GtkScale, GtkScaleButton, GtkScrollbar, GtkSeparator, GtkShortcutsGroup, GtkShortcutsSection, GtkShortcutsShortcut, GtkSpinButton, GtkStackSwitcher, GtkStatusbar, GtkToolPalette, GtkToolbar, GtkVBox, GtkVButtonBox, GtkVPaned, GtkVScale, GtkVScrollbar, GtkVSeparator and GtkVolumeButton.

Includes

```
#include <gtk/gtk.h>
```

Description

The GtkOrientable interface is implemented by all widgets that can be oriented horizontally or vertically. Historically, such widgets have been realized as subclasses of a common base class (e.g GtkBox/GtkHBox/GtkVBox or GtkScale/GtkHScale/GtkVScale). GtkOrientable is more flexible in that it allows the orientation to be changed at runtime, allowing the widgets to “flip”.

GtkOrientable was introduced in GTK+ 2.16.

Functions

gtk_orientable_get_orientation ()

```
GtkOrientation
gtk_orientable_get_orientation (GtkOrientable *orientable);
```

Retrieves the orientation of the `orientable` .

Parameters

orientable	a GtkOrientable

Returns

the orientation of the `orientable` .

Since: 2.16

gtk_orientable_set_orientation ()

```
void
gtk_orientable_set_orientation (GtkOrientable *orientable,
                                GtkOrientation orientation);
```

Sets the orientation of the `orientable`.

Parameters

orientable	a GtkOrientable
orientation	the orientable's new orientation.

Since: 2.16

Types and Values

GtkOrientable

```
typedef struct _GtkOrientable GtkOrientable;
```

Property Details

The `"orientation"` property

```
  "orientation"              GtkOrientation
```

The orientation of the orientable.

Flags: Read / Write

Default value: GTK_ORIENTATION_HORIZONTAL

Since: 2.16

GtkAspectFrame

GtkAspectFrame — A frame that constrains its child to a particular aspect ratio

Functions

GtkWidget *	gtk_aspect_frame_new ()
void	gtk_aspect_frame_set ()

Properties

gboolean	obey-child	Read / Write
gfloat	ratio	Read / Write
gfloat	xalign	Read / Write
gfloat	yalign	Read / Write

Types and Values

struct	GtkAspectFrame
struct	GtkAspectFrameClass

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkContainer
            └── GtkBin
                └── GtkFrame
                    └── GtkAspectFrame
```

Implemented Interfaces

GtkAspectFrame implements AtkImplementorIface and GtkBuildable.

Includes

```
#include <gtk/gtk.h>
```

Description

The GtkAspectFrame is useful when you want pack a widget so that it can resize but always retains the same aspect ratio. For instance, one might be drawing a small preview of a larger image. GtkAspectFrame derives from GtkFrame, so it can draw a label and a frame around the child. The frame will be “shrink-wrapped” to the size of the child.

CSS nodes

GtkAspectFrame uses a CSS node with name frame.

Functions

gtk_aspect_frame_new ()

```
GtkWidget *
gtk_aspect_frame_new (const gchar *label,
                      gfloat xalign,
                      gfloat yalign,
                      gfloat ratio,
                      gboolean obey_child);
```

Create a new GtkAspectFrame.

Parameters

label	Label text.	[allow-none]
xalign	Horizontal alignment of the child within the allocation of the GtkAspectFrame. This ranges from 0.0 (left aligned) to 1.0 (right aligned)	
yalign	Vertical alignment of the child within the allocation of the GtkAspectFrame. This ranges from 0.0 (top aligned) to 1.0 (bottom aligned)	
ratio	The desired aspect ratio.	
obey_child	If `TRUE`, `ratio` is ignored, and the aspect ratio is taken from the requistion of the child.	

Returns

the new GtkAspectFrame.

---

gtk_aspect_frame_set ()

```
void
gtk_aspect_frame_set (GtkAspectFrame *aspect_frame,
                      gfloat xalign,
                      gfloat yalign,
                      gfloat ratio,
                      gboolean obey_child);
```

Set parameters for an existing GtkAspectFrame.

Parameters

aspect_frame	a GtkAspectFrame
xalign	Horizontal alignment of the child within the allocation of the GtkAspectFrame. This ranges from 0.0 (left aligned) to 1.0 (right aligned)
yalign	Vertical alignment of the child within the allocation of the GtkAspectFrame. This ranges from 0.0 (top aligned) to 1.0 (bottom aligned)
ratio	The desired aspect ratio.
obey_child	If `TRUE`, `ratio` is ignored, and the aspect ratio is taken from the requistion of the child.

Types and Values

struct GtkAspectFrame

```
struct GtkAspectFrame;
```

---

struct GtkAspectFrameClass

```
struct GtkAspectFrameClass {
  GtkFrameClass parent_class;
};
```

Members

Property Details

The `"obey-child"` property

```
  "obey-child"               gboolean
```

Force aspect ratio to match that of the frame's child.

Flags: Read / Write

Default value: TRUE

The “ratio” property

“ratio” gfloat

Aspect ratio if obey_child is FALSE.

Flags: Read / Write

Allowed values: [0.0001,10000]

Default value: 1

The “xalign” property

“xalign” gfloat

X alignment of the child.

Flags: Read / Write

Allowed values: [0,1]

Default value: 0.5

The “yalign” property

“yalign” gfloat

Y alignment of the child.

Flags: Read / Write

Allowed values: [0,1]

Default value: 0.5

GtkFixed

GtkFixed — A container which allows you to position widgets at fixed coordinates

Functions

GtkWidget *	gtk_fixed_new ()
void	gtk_fixed_put ()
void	gtk_fixed_move ()

Child Properties

gint	x	Read / Write
gint	y	Read / Write

Types and Values

struct	GtkFixed

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkContainer
            └── GtkFixed
```

Implemented Interfaces

GtkFixed implements AtkImplementorIface and GtkBuildable.

Includes

```
#include <gtk/gtk.h>
```

Description

The GtkFixed widget is a container which can place child widgets at fixed positions and with fixed sizes, given in pixels. GtkFixed performs no automatic layout management.

For most applications, you should not use this container! It keeps you from having to learn about the other GTK+ containers, but it results in broken applications. With GtkFixed, the following things will result in truncated text, overlapping widgets, and other display bugs:

- Themes, which may change widget sizes.
- Fonts other than the one you used to write the app will of course change the size of widgets containing text; keep in mind that users may use a larger font because of difficulty reading the default, or they may be using a different OS that provides different fonts.
- Translation of text into other languages changes its size. Also, display of non-English text will use a different font in many cases.

In addition, GtkFixed does not pay attention to text direction and thus may produce unwanted results if your app is run under right-to-left languages such as Hebrew or Arabic. That is: normally GTK+ will order containers appropriately for the text direction, e.g. to put labels to the right of the thing they label when using an RTL language, but it can’t do that with GtkFixed. So if you need to reorder widgets depending on the text direction, you would need to manually detect it and adjust child positions accordingly.

Finally, fixed positioning makes it kind of annoying to add/remove GUI elements, since you have to reposition all the other elements. This is a long-term maintenance problem for your application.

If you know none of these things are an issue for your application, and prefer the simplicity of GtkFixed, by all means use the widget. But you should be aware of the tradeoffs.

See also GtkLayout, which shares the ability to perform fixed positioning of child widgets and additionally adds custom drawing and scrollability.

Functions

gtk_fixed_new ()

```
GtkWidget *
gtk_fixed_new (void);
```

Creates a new GtkFixed.

Returns

a new GtkFixed.

---

gtk_fixed_put ()

```
void
gtk_fixed_put (GtkFixed *fixed,
               GtkWidget *widget,
               gint x,
               gint y);
```

Adds a widget to a GtkFixed container at the given position.

Parameters

fixed	a GtkFixed.
widget	the widget to add.
x	the horizontal position to place the widget at.
y	the vertical position to place the widget at.

---

gtk_fixed_move ()

```
void
gtk_fixed_move (GtkFixed *fixed,
                GtkWidget *widget,
                gint x,
                gint y);
```

Moves a child of a GtkFixed container to the given position.

Parameters

fixed	a GtkFixed.
widget	the child widget.
x	the horizontal position to move the widget to.
y	the vertical position to move the

widget to.

Types and Values

struct GtkFixed

```
struct GtkFixed;
```

Child Property Details

The "x" child property

  "x"   gint

X position of child widget.

Flags: Read / Write

Default value: 0

The "y" child property

  "y"   gint

Y position of child widget.

Flags: Read / Write

Default value: 0

See Also

GtkLayout

Display Widgets

GtkLabel — A widget that displays a small to medium amount of text

GtkImage — A widget displaying an image

GtkSpinner — Show a spinner animation

GtkInfoBar — Report important messages to the user

GtkProgressBar — A widget which indicates progress visually

GtkLevelBar — A bar that can used as a level indicator

GtkStatusbar — Report messages of minor importance to the user

GtkAccelLabel — A label which displays an accelerator key on the right of the text

GtkLabel

GtkLabel — A widget that displays a small to medium amount of text

Functions

GtkWidget *	gtk_label_new ()
void	gtk_label_set_text ()
void	gtk_label_set_attributes ()
void	gtk_label_set_markup ()
void	gtk_label_set_markup_with_mnemonic ()
void	gtk_label_set_pattern ()
void	gtk_label_set_justify ()
void	gtk_label_set_xalign ()
void	gtk_label_set_yalign ()
void	gtk_label_set_ellipsize ()
void	gtk_label_set_width_chars ()
void	gtk_label_set_max_width_chars ()
void	gtk_label_set_line_wrap ()
void	gtk_label_set_line_wrap_mode ()
void	gtk_label_set_lines ()
void	gtk_label_get_layout_offsets ()
guint	gtk_label_get_mnemonic_keyval ()
gboolean	gtk_label_get_selectable ()
const gchar *	gtk_label_get_text ()
GtkWidget *	gtk_label_new_with_mnemonic ()
void	gtk_label_select_region ()
void	gtk_label_set_mnemonic_widget ()
void	gtk_label_set_selectable ()
void	gtk_label_set_text_with_mnemonic ()
PangoAttrList *	gtk_label_get_attributes ()
GtkJustification	gtk_label_get_justify ()
gfloat	gtk_label_get_xalign ()
gfloat	gtk_label_get_yalign ()
PangoEllipsizeMode	gtk_label_get_ellipsize ()
gint	gtk_label_get_width_chars ()
gint	gtk_label_get_max_width_chars ()
const gchar *	gtk_label_get_label ()
PangoLayout *	gtk_label_get_layout ()
gboolean	gtk_label_get_line_wrap ()
PangoWrapMode	gtk_label_get_line_wrap_mode ()
gint	gtk_label_get_lines ()

GtkWidget *	gtk_label_get_mnemonic_widget ()
gboolean	gtk_label_get_selection_bounds ()
gboolean	gtk_label_get_use_markup ()
gboolean	gtk_label_get_use_underline ()
gboolean	gtk_label_get_single_line_mode ()
gdouble	gtk_label_get_angle ()
void	gtk_label_set_label ()
void	gtk_label_set_use_markup ()
void	gtk_label_set_use_underline ()
void	gtk_label_set_single_line_mode ()
void	gtk_label_set_angle ()
const gchar *	gtk_label_get_current_uri ()
void	gtk_label_set_track_visited_links ()
gboolean	gtk_label_get_track_visited_links ()

Properties

gdouble	angle	Read / Write
PangoAttrList *	attributes	Read / Write
gint	cursor-position	Read
PangoEllipsizeMode	ellipsize	Read / Write
GtkJustification	justify	Read / Write
gchar *	label	Read / Write
gint	lines	Read / Write
gint	max-width-chars	Read / Write
guint	mnemonic-keyval	Read
GtkWidget *	mnemonic-widget	Read / Write
gchar *	pattern	Write
gboolean	selectable	Read / Write
gint	selection-bound	Read
gboolean	single-line-mode	Read / Write
gboolean	track-visited-links	Read / Write
gboolean	use-markup	Read / Write
gboolean	use-underline	Read / Write
gint	width-chars	Read / Write
gboolean	wrap	Read / Write
PangoWrapMode	wrap-mode	Read / Write
gfloat	xalign	Read / Write
gfloat	yalign	Read / Write

Signals

void	activate-current-link	Action
gboolean	activate-link	Run Last
void	copy-clipboard	Action
void	move-cursor	Action
void	populate-popup	Run Last

Types and Values

struct	GtkLabel

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkMisc
            └── GtkLabel
                └── GtkAccelLabel
```

Implemented Interfaces

GtkLabel implements AtkImplementorIface and GtkBuildable.

Includes

```
#include <gtk/gtk.h>
```

Description

The GtkLabel widget displays a small amount of text. As the name implies, most labels are used to label another widget such as a GtkButton, a GtkMenuItem, or a GtkComboBox.

CSS nodes

```
1   label
2   ├── [selection]
3   ├── [link]
4   ┊
5   └── [link]
```

GtkLabel has a single CSS node with the name label. A wide variety of style classes may be applied to labels, such as .title, .subtitle, .dim-label, etc. In the GtkShortcutsWindow, labels are used wth the .keycap style class.

If the label has a selection, it gets a subnode with name selection.

If the label has links, there is one subnode per link. These subnodes carry the link or visited state depending on whether they have been visited.

GtkLabel as GtkBuildable

The GtkLabel implementation of the GtkBuildable interface supports a custom <attributes> element, which supports any number of <attribute> elements. The <attribute> element has attributes named “name“, “value“, “start“ and “end“ and allows you to specify PangoAttribute values for this label.

An example of a UI definition fragment specifying Pango attributes:

```
1   <object class="GtkLabel">
2     <attributes>
3       <attribute name="weight" value="PANGO_WEIGHT_BOLD"/>
4       <attribute name="background" value="red" start="5" end="10"/>
5     </attributes>
6   </object>
```

The start and end attributes specify the range of characters to which the Pango attribute applies. If start and end are not specified, the attribute is applied to the whole text. Note that specifying ranges does not make much sense with translatable attributes. Use markup embedded in the translatable content instead.

Mnemonics

Labels may contain “mnemonics”. Mnemonics are underlined characters in the label, used for keyboard navigation. Mnemonics are created by providing a string with an underscore before the mnemonic character, such as `"_File"`, to the functions `gtk_label_new_with_mnemonic()` or `gtk_label_set_text_with_mnemonic()`.

Mnemonics automatically activate any activatable widget the label is inside, such as a GtkButton; if the label is not inside the mnemonic’s target widget, you have to tell the label about the target using `gtk_label_set_mnemonic_widget()`. Here’s a simple example where the label is inside a button:

```
1  // Pressing Alt+H will activate this button
2  GtkWidget *button = gtk_button_new ();
3  GtkWidget *label = gtk_label_new_with_mnemonic ("_Hello");
4  gtk_container_add (GTK_CONTAINER (button), label);
```

There’s a convenience function to create buttons with a mnemonic label already inside:

```
1  // Pressing Alt+H will activate this button
2  GtkWidget *button = gtk_button_new_with_mnemonic ("_Hello");
```

To create a mnemonic for a widget alongside the label, such as a GtkEntry, you have to point the label at the entry with `gtk_label_set_mnemonic_widget()`:

```
1  // Pressing Alt+H will focus the entry
2  GtkWidget *entry = gtk_entry_new ();
3  GtkWidget *label = gtk_label_new_with_mnemonic ("_Hello");
4  gtk_label_set_mnemonic_widget (GTK_LABEL (label), entry);
```

Markup (styled text)

To make it easy to format text in a label (changing colors, fonts, etc.), label text can be provided in a simple markup format.

Here's how to create a label with a small font:

```
1  GtkWidget *label = gtk_label_new (NULL);
2  gtk_label_set_markup (GTK_LABEL (label), "<small>Small text</small>");
```

(See complete documentation of available tags in the Pango manual.)

The markup passed to `gtk_label_set_markup()` must be valid; for example, literal <, > and & characters must be escaped as <, >, and &. If you pass text obtained from the user, file, or a network to `gtk_label_set_markup()`, you'll want to escape it with `g_markup_escape_text()` or `g_markup_printf_escaped()`.

Markup strings are just a convenient way to set the PangoAttrList on a label; `gtk_label_set_attributes()` may be a simpler way to set attributes in some cases. Be careful though; PangoAttrList tends to cause internationalization problems, unless you're applying attributes to the entire string (i.e. unless you set the range of each attribute to [0, `G_MAXINT`)). The reason is that specifying the start_index and end_index for a PangoAttribute requires knowledge of the exact string being displayed, so translations will cause problems.

Selectable labels

Labels can be made selectable with `gtk_label_set_selectable()`. Selectable labels allow the user to copy the label contents to the clipboard. Only labels that contain useful-to-copy information — such as error messages — should be made selectable.

Text layout

A label can contain any number of paragraphs, but will have performance problems if it contains more than a small number. Paragraphs are separated by newlines or other paragraph separators understood by Pango.

Labels can automatically wrap text if you call `gtk_label_set_line_wrap()`.

gtk_label_set_justify() sets how the lines in a label align with one another. If you want to set how the label as a whole aligns in its available space, see the "halign" and "valign" properties.

The "width-chars" and "max-width-chars" properties can be used to control the size allocation of ellipsized or wrapped labels. For ellipsizing labels, if either is specified (and less than the actual text size), it is used as the minimum width, and the actual text size is used as the natural width of the label. For wrapping labels, width-chars is used as the minimum width, if specified, and max-width-chars is used as the natural width. Even if max-width-chars specified, wrapping labels will be rewrapped to use all of the available width.

Note that the interpretation of "width-chars" and "max-width-chars" has changed a bit with the introduction of width-for-height geometry management.

Links

Since 2.18, GTK+ supports markup for clickable hyperlinks in addition to regular Pango markup. The markup for links is borrowed from HTML, using the <a> with “href“ and “title“ attributes. GTK+ renders links similar to the way they appear in web browsers, with colored, underlined text. The “title“ attribute is displayed as a tooltip on the link.

An example looks like this:

```
1   const gchar *text =
2   "Go to the"
3   "<a href=\"http://www.gtk.org title=\"<i>Our</i> website\">"
4   "GTK+ website</a> for more...";
5   GtkWidget *label = gtk_label_new (NULL);
6   gtk_label_set_markup (GTK_LABEL (label), text);
```

It is possible to implement custom handling for links and their tooltips with the “activate-link” signal and the `gtk_label_get_current_uri()` function.

Functions

gtk_label_new ()

```
GtkWidget *
gtk_label_new (const gchar *str);
```

Creates a new label with the given text inside it. You can pass `NULL` to get an empty label widget.

Parameters

str	The text of the label.	[nullable]

Returns

the new GtkLabel

---

gtk_label_set_text ()

```
void
gtk_label_set_text (GtkLabel *label,
                    const gchar *str);
```

Sets the text within the GtkLabel widget. It overwrites any text that was there before.

This function will clear any previously set mnemonic accelerators, and set the “use-underline” property to `FALSE` as a side effect.

This function will set the “use-markup” property to `FALSE` as a side effect.

See also: `gtk_label_set_markup()`

Parameters

label	a GtkLabel
str	The text you want to set

---

gtk_label_set_attributes ()

```
void
gtk_label_set_attributes (GtkLabel *label,
                          PangoAttrList *attrs);
```

Sets a PangoAttrList; the attributes in the list are applied to the label text.

The attributes set with this function will be applied and merged with any other attributes previously effected by way of the “use-underline” or “use-markup” properties. While it is not recommended to mix markup strings with manually set attributes, if you must; know that the attributes will be applied to the label after the markup string is parsed.

Parameters

label	a GtkLabel	
attrs	a PangoAttrList, or `NULL`.	[nullable]

---

gtk_label_set_markup ()

```
void
gtk_label_set_markup (GtkLabel *label,
                      const gchar *str);
```

Parses `str` which is marked up with the Pango text markup language, setting the label’s text and attribute list based on the parse results.

If the `str` is external data, you may need to escape it with `g_markup_escape_text()` or `g_markup_printf_escaped()`:

```
1   GtkWidget *label = gtk_label_new (NULL);
2   const char *str = "some text";
3   const char *format = "<span style=\"italic\">\%s</span>";
4   char *markup;
5
6   markup = g_markup_printf_escaped (format, str);
7   gtk_label_set_markup (GTK_LABEL (label), markup);
8   g_free (markup);
```

This function will set the “use-markup” property to `TRUE` as a side effect.

If you set the label contents using the “label” property you should also ensure that you set the “use-markup” property accordingly.

See also: `gtk_label_set_text()`

Parameters

label	a GtkLabel
str	a markup string (see Pango markup format)

---

gtk_label_set_markup_with_mnemonic ()

```
void
gtk_label_set_markup_with_mnemonic (GtkLabel *label,
                                    const gchar *str);
```

Parses `str` which is marked up with the Pango text markup language, setting the label's text and attribute list based on the parse results. If characters in `str` are preceded by an underscore, they are underlined indicating that they represent a keyboard accelerator called a mnemonic.

The mnemonic key can be used to activate another widget, chosen automatically, or explicitly using `gtk_label_set_mnemonic_widget()`.

Parameters

label	a GtkLabel
str	a markup string (see Pango markup format)

---

gtk_label_set_pattern ()

```
void
gtk_label_set_pattern (GtkLabel *label,
                       const gchar *pattern);
```

The pattern of underlines you want under the existing text within the GtkLabel widget. For example if the current text of the label says "FooBarBaz" passing a pattern of "___ ___" will underline "Foo" and "Baz" but not "Bar".

Parameters

label	The GtkLabel you want to set the pattern to.
pattern	The pattern as described above.

---

gtk_label_set_justify ()

```
void
gtk_label_set_justify (GtkLabel *label,
                       GtkJustification jtype);
```

Sets the alignment of the lines in the text of the label relative to each other. `GTK_JUSTIFY_LEFT` is the default value when the widget is first created with `gtk_label_new()`. If you instead want to set the alignment of the label as a whole, use `gtk_widget_set_halign()` instead. `gtk_label_set_justify()` has no effect on labels containing only a single line.

Parameters

label	a GtkLabel
jtype	a GtkJustification

---

gtk_label_set_xalign ()

```
void
gtk_label_set_xalign (GtkLabel *label,
                      gfloat xalign);
```

Sets the “xalign” property for `label`.

Parameters

label	a GtkLabel
xalign	the new xalign value, between 0 and 1

Since: 3.16

---

gtk_label_set_yalign ()

```
void
gtk_label_set_yalign (GtkLabel *label,
                      gfloat yalign);
```

Sets the “yalign” property for `label`.

Parameters

label	a GtkLabel
yalign	the new yalign value, between 0 and 1

Since: 3.16

---

gtk_label_set_ellipsize ()

```
void
gtk_label_set_ellipsize (GtkLabel *label,
                         PangoEllipsizeMode mode);
```

Sets the mode used to ellipsize (add an ellipsis: "...") to the text if there is not enough space to render the entire string.

Parameters

label	a GtkLabel
mode	a PangoEllipsizeMode

Since: 2.6

---

gtk_label_set_width_chars ()

```
void
gtk_label_set_width_chars (GtkLabel *label,
                           gint n_chars);
```

Sets the desired width in characters of `label` to `n_chars`.

Parameters

label	a GtkLabel
n_chars	the new desired width, in characters.

Since: 2.6

---

gtk_label_set_max_width_chars ()

```
void
gtk_label_set_max_width_chars (GtkLabel *label,
                               gint n_chars);
```

Sets the desired maximum width in characters of `label` to `n_chars` .

Parameters

label	a GtkLabel
n_chars	the new desired maximum width, in characters.

Since: 2.6

---

gtk_label_set_line_wrap ()

```
void
gtk_label_set_line_wrap (GtkLabel *label,
                         gboolean wrap);
```

Toggles line wrapping within the GtkLabel widget. `TRUE` makes it break lines if text exceeds the widget's size. `FALSE` lets the text get cut off by the edge of the widget if it exceeds the widget size.

Note that setting line wrapping to `TRUE` does not make the label wrap at its parent container's width, because GTK+ widgets conceptually can't make their requisition depend on the parent container's size. For a label that wraps at a specific position, set the label's width using `gtk_widget_set_size_request()`.

Parameters

label	a GtkLabel
wrap	the setting

---

gtk_label_set_line_wrap_mode ()

```
void
gtk_label_set_line_wrap_mode (GtkLabel *label,
                              PangoWrapMode wrap_mode);
```

If line wrapping is on (see `gtk_label_set_line_wrap()`) this controls how the line wrapping is done. The default is `PANGO_WRAP_WORD` which means wrap on word boundaries.

Parameters

label	a GtkLabel
wrap_mode	the line wrapping mode

Since: 2.10

---

gtk_label_set_lines ()

```
void
gtk_label_set_lines (GtkLabel *label,
                     gint lines);
```

Sets the number of lines to which an ellipsized, wrapping label should be limited. This has no effect if the label is not wrapping or ellipsized. Set this to -1 if you don’t want to limit the number of lines.

Parameters

label	a GtkLabel
lines	the desired number of lines, or -1

Since: 3.10

---

gtk_label_get_layout_offsets ()

```
void
gtk_label_get_layout_offsets (GtkLabel *label,
                              gint *x,
                              gint *y);
```

Obtains the coordinates where the label will draw the PangoLayout representing the text in the label; useful to convert mouse events into coordinates inside the PangoLayout, e.g. to take some action if some part of the label is clicked. Of course you will need to create a GtkEventBox to receive the events, and pack the label inside it, since labels are windowless (they return `FALSE` from `gtk_widget_get_has_window()`). Remember when using the PangoLayout functions you need to convert to and from pixels using `PANGO_PIXELS()` or PANGO_SCALE.

Parameters

label	a GtkLabel	
x	location to store X offset of layout, or `NULL`.	[out][optional]
y	location to store Y offset of layout, or `NULL`.	[out][optional]

---

gtk_label_get_mnemonic_keyval ()

```
guint
gtk_label_get_mnemonic_keyval (GtkLabel *label);
```

If the label has been set so that it has an mnemonic key this function returns the keyval used for the mnemonic accelerator. If there is no mnemonic set up it returns GDK_KEY_VoidSymbol.

Parameters

label	a GtkLabel

Returns

GDK keyval usable for accelerators, or GDK_KEY_VoidSymbol

---

gtk_label_get_selectable ()

```
gboolean
gtk_label_get_selectable (GtkLabel *label);
```

Gets the value set by `gtk_label_set_selectable()`.

Parameters

label	a GtkLabel

Returns

`TRUE` if the user can copy text from the label

---

gtk_label_get_text ()

```
const gchar *
gtk_label_get_text (GtkLabel *label);
```

Fetches the text from a label widget, as displayed on the screen. This does not include any embedded underlines indicating mnemonics or Pango markup. (See `gtk_label_get_label()`)

Parameters

label	a GtkLabel

Returns

the text in the label widget. This is the internal string used by the label, and must not be modified.

---

gtk_label_new_with_mnemonic ()

```
GtkWidget *
gtk_label_new_with_mnemonic (const gchar *str);
```

Creates a new GtkLabel, containing the text in `str` .

If characters in `str` are preceded by an underscore, they are underlined. If you need a literal underscore character in a label, use '__' (two underscores). The first underlined character represents a keyboard accelerator called a mnemonic. The mnemonic key can be used to activate another widget, chosen automatically, or explicitly using `gtk_label_set_mnemonic_widget()`.

If `gtk_label_set_mnemonic_widget()` is not called, then the first activatable ancestor of the GtkLabel will be chosen as the mnemonic widget. For instance, if the label is inside a button or menu item, the button or menu item will automatically become the mnemonic widget and be activated by the mnemonic.

Parameters

str	The text of the label, with an underscore in front of the mnemonic character. [nullable]

Returns

the new GtkLabel

---

gtk_label_select_region ()

```
void
gtk_label_select_region (GtkLabel *label,
                         gint start_offset,
                         gint end_offset);
```

Selects a range of characters in the label, if the label is selectable. See `gtk_label_set_selectable()`. If the label is not selectable, this function has no effect. If `start_offset` or `end_offset` are -1, then the end of the label will be substituted.

Parameters

label	a GtkLabel	
start_offset	start offset (in characters not bytes)	
end_offset	end offset (in characters not bytes)	

---

gtk_label_set_mnemonic_widget ()

```
void
gtk_label_set_mnemonic_widget (GtkLabel *label,
                               GtkWidget *widget);
```

If the label has been set so that it has an mnemonic key (using i.e. `gtk_label_set_markup_with_mnemonic()`, `gtk_label_set_text_with_mnemonic()`, `gtk_label_new_with_mnemonic()` or the "use_underline" property) the label can be associated with a widget that is the target of the mnemonic. When the label is inside a widget (like a GtkButton or a GtkNotebook tab) it is automatically associated with the correct widget, but sometimes (i.e. when the target is a GtkEntry next to the label) you need to set it explicitly using this function.

The target widget will be accelerated by emitting the GtkWidget::mnemonic-activate signal on it. The default handler for this signal will activate the widget if there are no mnemonic collisions and toggle focus between the colliding widgets otherwise.

Parameters

label	a GtkLabel	
widget	the target GtkWidget, or `NULL` to unset.	[nullable]

---

gtk_label_set_selectable ()

```
void
gtk_label_set_selectable (GtkLabel *label,
                          gboolean setting);
```

Selectable labels allow the user to select text from the label, for copy-and-paste.

Parameters

label	a GtkLabel
setting	`TRUE` to allow selecting text in the label

---

gtk_label_set_text_with_mnemonic ()

```
void
gtk_label_set_text_with_mnemonic (GtkLabel *label,
                                  const gchar *str);
```

Sets the label's text from the string `str` . If characters in `str` are preceded by an underscore, they are underlined indicating that they represent a keyboard accelerator called a mnemonic. The mnemonic key can be used to activate another widget, chosen automatically, or explicitly using `gtk_label_set_mnemonic_widget()`.

Parameters

label	a GtkLabel
str	a string

---

gtk_label_get_attributes ()

```
PangoAttrList *
gtk_label_get_attributes (GtkLabel *label);
```

Gets the attribute list that was set on the label using `gtk_label_set_attributes()`, if any. This function does not reflect attributes that come from the labels markup (see `gtk_label_set_markup()`). If you want to get the effective attributes for the label, use pango_layout_get_attribute (gtk_label_get_layout (label)).

Parameters

label	a GtkLabel

Returns

the attribute list, or `NULL` if none was set.

[nullable][transfer none]

---

gtk_label_get_justify ()

```
GtkJustification
gtk_label_get_justify (GtkLabel *label);
```

Returns the justification of the label. See `gtk_label_set_justify()`.

Parameters

label	a GtkLabel

Returns

GtkJustification

---

gtk_label_get_xalign ()

```
gfloat
gtk_label_get_xalign (GtkLabel *label);
```

Gets the “xalign” property for `label`.

Parameters

label	a GtkLabel

Returns

the xalign property

Since: 3.16

---

gtk_label_get_yalign ()

```
gfloat
gtk_label_get_yalign (GtkLabel *label);
```

Gets the “yalign” property for `label`.

Parameters

label	a GtkLabel

Returns

the yalign property

Since: 3.16

---

gtk_label_get_ellipsize ()

```
PangoEllipsizeMode
gtk_label_get_ellipsize (GtkLabel *label);
```

Returns the ellipsizing position of the label. See `gtk_label_set_ellipsize()`.

Parameters

label	a GtkLabel

Returns

PangoEllipsizeMode

Since: 2.6

---

gtk_label_get_width_chars ()

```
gint
gtk_label_get_width_chars (GtkLabel *label);
```

Retrieves the desired width of `label` , in characters. See `gtk_label_set_width_chars()`.

Parameters

label	a GtkLabel

Returns

the width of the label in characters.

Since: 2.6

---

gtk_label_get_max_width_chars ()

```
gint
gtk_label_get_max_width_chars (GtkLabel *label);
```

Retrieves the desired maximum width of `label` , in characters. See `gtk_label_set_width_chars()`.

Parameters

label	a GtkLabel

Returns

the maximum width of the label in characters.

Since: 2.6

---

gtk_label_get_label ()

```
const gchar *
gtk_label_get_label (GtkLabel *label);
```

Fetches the text from a label widget including any embedded underlines indicating mnemonics and Pango markup. (See `gtk_label_get_text()`).

Parameters

label	a GtkLabel

Returns

the text of the label widget. This string is owned by the widget and must not be modified or freed.

---

gtk_label_get_layout ()

```
PangoLayout *
gtk_label_get_layout (GtkLabel *label);
```

Gets the PangoLayout used to display the label. The layout is useful to e.g. convert text positions to pixel positions, in combination with `gtk_label_get_layout_offsets()`. The returned layout is owned by the `label` so need not be freed by the caller. The `label` is free to recreate its layout at any time, so it should be considered read-only.

Parameters

label	a GtkLabel

Returns

the PangoLayout for this label.

[transfer none]

---

gtk_label_get_line_wrap ()

```
gboolean
gtk_label_get_line_wrap (GtkLabel *label);
```

Returns whether lines in the label are automatically wrapped. See `gtk_label_set_line_wrap()`.

Parameters

label	a GtkLabel

Returns

`TRUE` if the lines of the label are automatically wrapped.

---

gtk_label_get_line_wrap_mode ()

```
PangoWrapMode
gtk_label_get_line_wrap_mode (GtkLabel *label);
```

Returns line wrap mode used by the label. See `gtk_label_set_line_wrap_mode()`.

Parameters

label	a GtkLabel

Returns

`TRUE` if the lines of the label are automatically wrapped.

Since: 2.10

---

gtk_label_get_lines ()

```
gint
gtk_label_get_lines (GtkLabel *label);
```

Gets the number of lines to which an ellipsized, wrapping label should be limited. See `gtk_label_set_lines()`.

Parameters

label	a GtkLabel

Returns

The number of lines

Since: 3.10

---

gtk_label_get_mnemonic_widget ()

```
GtkWidget *
gtk_label_get_mnemonic_widget (GtkLabel *label);
```

Retrieves the target of the mnemonic (keyboard shortcut) of this label. See `gtk_label_set_mnemonic_widget()`.

Parameters

label	a GtkLabel

Returns

the target of the label's mnemonic, or `NULL` if none has been set and the default algorithm will be used.

[nullable][transfer none]

---

gtk_label_get_selection_bounds ()

```
gboolean
gtk_label_get_selection_bounds (GtkLabel *label,
                                gint *start,
                                gint *end);
```

Gets the selected range of characters in the label, returning `TRUE` if there's a selection.

Parameters

label	a GtkLabel	
start	return location for start of selection, as a character offset.	[out]
end	return location for end of selection, as a character offset.	[out]

Returns

`TRUE` if selection is non-empty

---

gtk_label_get_use_markup ()

```
gboolean
gtk_label_get_use_markup (GtkLabel *label);
```

Returns whether the label's text is interpreted as marked up with the Pango text markup language. See `gtk_label_set_use_markup()`.

Parameters

label	a GtkLabel

Returns

`TRUE` if the label's text will be parsed for markup.

---

gtk_label_get_use_underline ()

```
gboolean
gtk_label_get_use_underline (GtkLabel *label);
```

Returns whether an embedded underline in the label indicates a mnemonic. See `gtk_label_set_use_underline()`.

Parameters

label	a GtkLabel

Returns

`TRUE` whether an embedded underline in the label indicates the mnemonic accelerator keys.

---

gtk_label_get_single_line_mode ()

```
gboolean
gtk_label_get_single_line_mode (GtkLabel *label);
```

Returns whether the label is in single line mode.

Parameters

label	a GtkLabel

Returns

`TRUE` when the label is in single line mode.

Since: 2.6

---

gtk_label_get_angle ()

```
gdouble
gtk_label_get_angle (GtkLabel *label);
```

Gets the angle of rotation for the label. See `gtk_label_set_angle()`.

Parameters

label	a GtkLabel

Returns

the angle of rotation for the label

Since: 2.6

---

gtk_label_set_label ()

```
void
gtk_label_set_label (GtkLabel *label,
                     const gchar *str);
```

Sets the text of the label. The label is interpreted as including embedded underlines and/or Pango markup depending on the values of the “use-underline” and “use-markup” properties.

Parameters

label	a GtkLabel
str	the new text to set for the label

---

gtk_label_set_use_markup ()

```
void
gtk_label_set_use_markup (GtkLabel *label,
                          gboolean setting);
```

Sets whether the text of the label contains markup in Pango’s text markup language. See `gtk_label_set_markup()`.

Parameters

label	a GtkLabel
setting	`TRUE` if the label’s text should be parsed for markup.

---

gtk_label_set_use_underline ()

```
void
gtk_label_set_use_underline (GtkLabel *label,
                             gboolean setting);
```

If true, an underline in the text indicates the next character should be used for the mnemonic accelerator key.

Parameters

label	a GtkLabel
setting	TRUE if underlines in the text indicate mnemonics

---

gtk_label_set_single_line_mode ()

```
void
gtk_label_set_single_line_mode (GtkLabel *label,
                                gboolean single_line_mode);
```

Sets whether the label is in single line mode.

Parameters

label	a GtkLabel
single_line_mode	TRUE if the label should be in single line mode

Since: 2.6

---

gtk_label_set_angle ()

```
void
gtk_label_set_angle (GtkLabel *label,
                     gdouble angle);
```

Sets the angle of rotation for the label. An angle of 90 reads from from bottom to top, an angle of 270, from top to bottom. The angle setting for the label is ignored if the label is selectable, wrapped, or ellipsized.

Parameters

label	a GtkLabel
angle	the angle that the baseline of the label makes with the horizontal, in degrees, measured counterclockwise

Since: 2.6

---

gtk_label_get_current_uri ()

```
const gchar *
gtk_label_get_current_uri (GtkLabel *label);
```

Returns the URI for the currently active link in the label. The active link is the one under the mouse pointer or, in a selectable label, the link in which the text cursor is currently positioned.

This function is intended for use in a “activate-link” handler or for use in a “query-tooltip” handler.

Parameters

label	a GtkLabel

Returns

the currently active URI. The string is owned by GTK+ and must not be freed or modified.

Since: 2.18

---

gtk_label_set_track_visited_links ()

```
void
gtk_label_set_track_visited_links (GtkLabel *label,
                                   gboolean track_links);
```

Sets whether the label should keep track of clicked links (and use a different color for them).

Parameters

label	a GtkLabel
track_links	TRUE to track visited links

Since: 2.18

---

gtk_label_get_track_visited_links ()

```
gboolean
gtk_label_get_track_visited_links (GtkLabel *label);
```

Returns whether the label is currently keeping track of clicked links.

Parameters

label	a GtkLabel

Returns

TRUE if clicked links are remembered

Since: 2.18

Types and Values

struct GtkLabel

```
struct GtkLabel;
```

Property Details

The “angle” property

```
  “angle”                    gdouble
```

The angle that the baseline of the label makes with the horizontal, in degrees, measured counterclockwise. An angle of 90 reads from from bottom to top, an angle of 270, from top to bottom. Ignored if the label is selectable.

Flags: Read / Write

Allowed values: [0,360]

Default value: 0

Since: 2.6

The “attributes” property

```
  “attributes”               PangoAttrList *
```

A list of style attributes to apply to the text of the label.

Flags: Read / Write

The “cursor-position” property

```
  “cursor-position”          gint
```

The current position of the insertion cursor in chars.

Flags: Read

Allowed values: >= 0

Default value: 0

The “ellipsize” property

```
  “ellipsize”                PangoEllipsizeMode
```

The preferred place to ellipsize the string, if the label does not have enough room to display the entire string, specified as a PangoEllipsizeMode.

Note that setting this property to a value other than `PANGO_ELLIPSIZE_NONE` has the side-effect that the label requests only enough space to display the ellipsis "...". In particular, this means that ellipsizing labels do not work well in notebook tabs, unless the GtkNotebook tab-expand child property is set to `TRUE`. Other ways to set a label's width are `gtk_widget_set_size_request()` and `gtk_label_set_width_chars()`.

Flags: Read / Write

Default value: PANGO_ELLIPSIZE_NONE

Since: 2.6

The `"justify"` property

```
"justify"                  GtkJustification
```

The alignment of the lines in the text of the label relative to each other. This does NOT affect the alignment of the label within its allocation. See GtkLabel:xalign for that.

Flags: Read / Write

Default value: GTK_JUSTIFY_LEFT

The `"label"` property

```
"label"                    gchar *
```

The contents of the label.

If the string contains Pango XML markup, you will have to set the "use-markup" property to `TRUE` in order for the label to display the markup attributes. See also `gtk_label_set_markup()` for a convenience function that sets both this property and the "use-markup" property at the same time.

If the string contains underlines acting as mnemonics, you will have to set the "use-underline" property to `TRUE` in order for the label to display them.

Flags: Read / Write

Default value: ""

The `"lines"` property

```
"lines"                    gint
```

The number of lines to which an ellipsized, wrapping label should be limited. This property has no effect if the label is not wrapping or ellipsized. Set this property to -1 if you don't want to limit the number of lines.

Flags: Read / Write

Allowed values: >= -1

Default value: -1

Since: 3.10

The “max-width-chars” property

“max-width-chars” gint

The desired maximum width of the label, in characters. If this property is set to -1, the width will be calculated automatically.

See the section on text layout for details of how “width-chars” and “max-width-chars” determine the width of ellipsized and wrapped labels.

Flags: Read / Write

Allowed values: >= -1

Default value: -1

Since: 2.6

The “mnemonic-keyval” property

“mnemonic-keyval” guint

The mnemonic accelerator key for this label.

Flags: Read

Default value: 16777215

The “mnemonic-widget” property

“mnemonic-widget” GtkWidget *

The widget to be activated when the label's mnemonic key is pressed.

Flags: Read / Write

The “pattern” property

“pattern” gchar *

A string with _ characters in positions correspond to characters in the text to underline.

Flags: Write

Default value: NULL

The “selectable” property

“selectable” gboolean

Whether the label text can be selected with the mouse.

Flags: Read / Write

Default value: FALSE

The “selection-bound” property

“selection-bound” gint

The position of the opposite end of the selection from the cursor in chars.

Flags: Read

Allowed values: >= 0

Default value: 0

The “single-line-mode” property

“single-line-mode” gboolean

Whether the label is in single line mode. In single line mode, the height of the label does not depend on the actual text, it is always set to ascent + descent of the font. This can be an advantage in situations where resizing the label because of text changes would be distracting, e.g. in a statusbar.

Flags: Read / Write

Default value: FALSE

Since: 2.6

The “track-visited-links” property

“track-visited-links” gboolean

Set this property to TRUE to make the label track which links have been visited. It will then apply the GTK_STATE_FLAG_VISITED when rendering this link, in addition to GTK_STATE_FLAG_LINK.

Flags: Read / Write

Default value: TRUE

Since: 2.18

The “use-markup” property

“use-markup” gboolean

The text of the label includes XML markup. See pango_parse_markup().

Flags: Read / Write

Default value: FALSE

The “use-underline” property

“use-underline” gboolean

If set, an underline in the text indicates the next character should be used for the mnemonic accelerator key.

Flags: Read / Write

Default value: FALSE

The “width-chars” property

“width-chars” gint

The desired width of the label, in characters. If this property is set to -1, the width will be calculated automatically.

See the section on text layout for details of how “width-chars” and “max-width-chars” determine the width of ellipsized and wrapped labels.

Flags: Read / Write

Allowed values: >= -1

Default value: -1

Since: 2.6

The “wrap” property

“wrap” gboolean

If set, wrap lines if the text becomes too wide.

Flags: Read / Write

Default value: FALSE

The “wrap-mode” property

“wrap-mode” PangoWrapMode

If line wrapping is on (see the “wrap” property) this controls how the line wrapping is done. The default is PANGO_WRAP_WORD, which means wrap on word boundaries.

Flags: Read / Write

Default value: PANGO_WRAP_WORD

Since: 2.10

The “xalign” property

```
  “xalign”                   gfloat
```

The xalign property determines the horizontal aligment of the label text inside the labels size allocation. Compare this to “halign”, which determines how the labels size allocation is positioned in the space available for the label.

Flags: Read / Write

Allowed values: [0,1]

Default value: 0.5

Since: 3.16

The “yalign” property

```
  “yalign”                   gfloat
```

The yalign property determines the vertical aligment of the label text inside the labels size allocation. Compare this to “valign”, which determines how the labels size allocation is positioned in the space available for the label.

Flags: Read / Write

Allowed values: [0,1]

Default value: 0.5

Since: 3.16

Signal Details

The “activate-current-link” signal

```
void
user_function (GtkLabel *label,
               gpointer  user_data)
```

A keybinding signal which gets emitted when the user activates a link in the label.

Applications may also emit the signal with `g_signal_emit_by_name()` if they need to control activation of URIs programmatically.

The default bindings for this signal are all forms of the Enter key.

Parameters

label	The label on which the signal was emitted
user_data	user data set when the signal handler was connected.

Flags: Action

Since: 2.18

The `"activate-link"` signal

```
gboolean
user_function (GtkLabel *label,
               gchar    *uri,
               gpointer  user_data)
```

The signal which gets emitted to activate a URI. Applications may connect to it to override the default behaviour, which is to call `gtk_show_uri_on_window()`.

Parameters

label	The label on which the signal was emitted
uri	the URI that is activated
user_data	user data set when the signal handler was connected.

Returns

`TRUE` if the link has been activated

Flags: Run Last

Since: 2.18

The `"copy-clipboard"` signal

```
void
user_function (GtkLabel *label,
               gpointer  user_data)
```

The ::copy-clipboard signal is a keybinding signal which gets emitted to copy the selection to the clipboard.

The default binding for this signal is Ctrl-c.

Parameters

label	the object which received the signal
user_data	user data set when the signal handler was connected.

Flags: Action

The “move-cursor” signal

```
void
user_function (GtkLabel       *entry,
               GtkMovementStep step,
               gint            count,
               gboolean        extend_selection,
               gpointer        user_data)
```

The ::move-cursor signal is a keybinding signal which gets emitted when the user initiates a cursor movement. If the cursor is not visible in `entry` , this signal causes the viewport to be moved instead.

Applications should not connect to it, but may emit it with `g_signal_emit_by_name()` if they need to control the cursor programmatically.

The default bindings for this signal come in two variants, the variant with the Shift modifier extends the selection, the variant without the Shift modifer does not. There are too many key combinations to list them all here.

- Arrow keys move by individual characters/lines
- Ctrl-arrow key combinations move by words/paragraphs
- Home/End keys move to the ends of the buffer

Parameters

entry	the object which received the signal
step	the granularity of the move, as a GtkMovementStep
count	the number of `step` units to move
extend_selection	`TRUE` if the move should extend the selection
user_data	user data set when the signal handler was connected.

Flags: Action

The “populate-popup” signal

```
void
user_function (GtkLabel *label,
               GtkMenu  *menu,
               gpointer  user_data)
```

The ::populate-popup signal gets emitted before showing the context menu of the label. Note that only selectable labels have context menus.

If you need to add items to the context menu, connect to this signal and append your menuitems to the `menu` .

Parameters

label	The label on which the signal is emitted
menu	the menu that is being populated
user_data	user data set when the signal handler was connected.

Flags: Run Last

GtkImage

GtkImage — A widget displaying an image

Functions

void	gtk_image_get_icon_set ()
GdkPixbuf *	gtk_image_get_pixbuf ()
void	gtk_image_get_stock ()
GdkPixbufAnimation *	gtk_image_get_animation ()
void	gtk_image_get_icon_name ()
void	gtk_image_get_gicon ()
GtkImageType	gtk_image_get_storage_type ()
GtkWidget *	gtk_image_new_from_file ()
GtkWidget *	gtk_image_new_from_icon_set ()
GtkWidget *	gtk_image_new_from_pixbuf ()
GtkWidget *	gtk_image_new_from_stock ()
GtkWidget *	gtk_image_new_from_animation ()
GtkWidget *	gtk_image_new_from_icon_name ()
GtkWidget *	gtk_image_new_from_gicon ()
GtkWidget *	gtk_image_new_from_resource ()
GtkWidget *	gtk_image_new_from_surface ()
void	gtk_image_set_from_file ()
void	gtk_image_set_from_icon_set ()
void	gtk_image_set_from_pixbuf ()
void	gtk_image_set_from_stock ()
void	gtk_image_set_from_animation ()
void	gtk_image_set_from_icon_name ()
void	gtk_image_set_from_gicon ()
void	gtk_image_set_from_resource ()
void	gtk_image_set_from_surface ()
void	gtk_image_clear ()
GtkWidget *	gtk_image_new ()

void	gtk_image_set_pixel_size ()
gint	gtk_image_get_pixel_size ()

Properties

gchar *	file	Read / Write
GIcon *	gicon	Read / Write
gchar *	icon-name	Read / Write
GtkIconSet *	icon-set	Read / Write
gint	icon-size	Read / Write
GdkPixbuf *	pixbuf	Read / Write
GdkPixbufAnimation *	pixbuf-animation	Read / Write
gint	pixel-size	Read / Write
gchar *	resource	Read / Write
gchar *	stock	Read / Write
GtkImageType	storage-type	Read
CairoSurface *	surface	Read / Write
gboolean	use-fallback	Read / Write

Types and Values

struct	GtkImage
enum	GtkImageType

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkMisc
            └── GtkImage
```

Implemented Interfaces

GtkImage implements AtkImplementorIface and GtkBuildable.

Includes

```
#include <gtk/gtk.h>
```

Description

The GtkImage widget displays an image. Various kinds of object can be displayed as an image; most typically, you would load a GdkPixbuf ("pixel buffer") from a file, and then display that. There's a convenience function to do this, `gtk_image_new_from_file()`, used as follows:

```
1 GtkWidget *image;
2 image = gtk_image_new_from_file
  ("myfile.png");
```

If the file isn't loaded successfully, the image will contain a "broken image" icon similar to that used in many web browsers. If you want to handle errors in loading the file yourself, for example by displaying an error

message, then load the image with gdk_pixbuf_new_from_file(), then create the GtkImage with gtk_image_new_from_pixbuf().

The image file may contain an animation, if so the GtkImage will display an animation (GdkPixbufAnimation) instead of a static image.

GtkImage is a subclass of GtkMisc, which implies that you can align it (center, left, right) and add padding to it, using GtkMisc methods.

GtkImage is a “no window” widget (has no GdkWindow of its own), so by default does not receive events. If you want to receive events on the image, such as button clicks, place the image inside a GtkEventBox, then connect to the event signals on the event box.

Handling button press events on a GtkImage.

```
1   static gboolean
2   button_press_callback (GtkWidget      *event_box,
3                          GdkEventButton *event,
4                          gpointer        data)
5   {
6     g_print ("Event box clicked at coordinates %f,%f\n",
7              event->x, event->y);
8
9     // Returning TRUE means we handled the event, so the signal
10    // emission should be stopped (don’t call any further callbacks
11    // that may be connected). Return FALSE to continue invoking callbacks.
12    return TRUE;
13  }
14
15  static GtkWidget*
16  create_image (void)
17  {
18    GtkWidget *image;
19    GtkWidget *event_box;
20
21    image = gtk_image_new_from_file ("myfile.png");
22
23    event_box = gtk_event_box_new ();
24
25    gtk_container_add (GTK_CONTAINER (event_box), image);
26
27    g_signal_connect (G_OBJECT (event_box),
28                      "button_press_event",
29                      G_CALLBACK (button_press_callback),
30                      image);
31
32    return image;
33  }
```

When handling events on the event box, keep in mind that coordinates in the image may be different from event box coordinates due to the alignment and padding settings on the image (see GtkMisc). The simplest way to solve this is to set the alignment to 0.0 (left/top), and set the padding to zero. Then the origin of the image will be the same as the origin of the event box.

Sometimes an application will want to avoid depending on external data files, such as image files. GTK+ comes with a program to avoid this, called “gdk-pixbuf-csource”. This library allows you to convert an image into a C variable declaration, which can then be loaded into a GdkPixbuf using gdk_pixbuf_new_from_inline().

CSS nodes

GtkImage has a single CSS node with the name image. The style classes may appear on image CSS nodes: .icon-dropshadow, .lowres-icon.

Functions

gtk_image_get_icon_set ()

```
void
gtk_image_get_icon_set (GtkImage *image,
                        GtkIconSet **icon_set,
                        GtkIconSize *size);
```

`gtk_image_get_icon_set` has been deprecated since version 3.10 and should not be used in newly-written code.

Use `gtk_image_get_icon_name()` instead.

Gets the icon set and size being displayed by the GtkImage. The storage type of the image must be `GTK_IMAGE_EMPTY` or `GTK_IMAGE_ICON_SET` (see `gtk_image_get_storage_type()`).

Parameters

image	a GtkImage	
icon_set	location to store a GtkIconSet, or `NULL`.	[out][transfer none][allow-none]
size	location to store a stock icon size (GtkIconSize), or `NULL`.	[out][allow-none][type int]

---

gtk_image_get_pixbuf ()

```
GdkPixbuf *
gtk_image_get_pixbuf (GtkImage *image);
```

Gets the GdkPixbuf being displayed by the GtkImage. The storage type of the image must be `GTK_IMAGE_EMPTY` or `GTK_IMAGE_PIXBUF` (see `gtk_image_get_storage_type()`). The caller of this function does not own a reference to the returned pixbuf.

Parameters

image	a GtkImage

Returns

the displayed pixbuf, or `NULL` if the image is empty.

[nullable][transfer none]

---

gtk_image_get_stock ()

```
void
gtk_image_get_stock (GtkImage *image,
                     gchar **stock_id,
                     GtkIconSize *size);
```

`gtk_image_get_stock` has been deprecated since version 3.10 and should not be used in newly-written code.

Use `gtk_image_get_icon_name()` instead.

Gets the stock icon name and size being displayed by the GtkImage. The storage type of the image must be `GTK_IMAGE_EMPTY` or `GTK_IMAGE_STOCK` (see `gtk_image_get_storage_type()`). The returned string is owned by the GtkImage and should not be freed.

Parameters

image	a GtkImage	
stock_id	place to store a stock icon name, or `NULL`.	[out][transfer none][allow-none]
size	place to store a stock icon size (GtkIconSize), or `NULL`.	[out][allow-none][type int]

---

gtk_image_get_animation ()

```
GdkPixbufAnimation *
gtk_image_get_animation (GtkImage *image);
```

Gets the GdkPixbufAnimation being displayed by the GtkImage. The storage type of the image must be `GTK_IMAGE_EMPTY` or `GTK_IMAGE_ANIMATION` (see `gtk_image_get_storage_type()`). The caller of this function does not own a reference to the returned animation.

Parameters

image	a GtkImage

Returns

the displayed animation, or `NULL` if the image is empty.

[nullable][transfer none]

---

gtk_image_get_icon_name ()

```
void
gtk_image_get_icon_name (GtkImage *image,
                         const gchar **icon_name,
                         GtkIconSize *size);
```

Gets the icon name and size being displayed by the GtkImage. The storage type of the image must be

GTK_IMAGE_EMPTY or GTK_IMAGE_ICON_NAME (see gtk_image_get_storage_type()). The returned string is owned by the GtkImage and should not be freed.

Parameters

image	a GtkImage	
icon_name	place to store an icon name, or NULL.	[out][transfer none][allow-none]
size	place to store an icon size (GtkIconSize), or NULL.	[out][allow-none][type int]

Since: 2.6

---

gtk_image_get_gicon ()

```
void
gtk_image_get_gicon (GtkImage *image,
                     GIcon **gicon,
                     GtkIconSize *size);
```

Gets the GIcon and size being displayed by the GtkImage. The storage type of the image must be GTK_IMAGE_EMPTY or GTK_IMAGE_GICON (see gtk_image_get_storage_type()). The caller of this function does not own a reference to the returned GIcon.

Parameters

image	a GtkImage	
gicon	place to store a GIcon, or NULL.	[out][transfer none][allow-none]
size	place to store an icon size (GtkIconSize), or NULL.	[out][allow-none][type int]

Since: 2.14

---

gtk_image_get_storage_type ()

```
GtkImageType
gtk_image_get_storage_type (GtkImage *image);
```

Gets the type of representation being used by the GtkImage to store image data. If the GtkImage has no image data, the return value will be GTK_IMAGE_EMPTY.

Parameters

image	a GtkImage

Returns

image representation being used

---

gtk_image_new_from_file ()

```
GtkWidget *
gtk_image_new_from_file (const gchar *filename);
```

Creates a new GtkImage displaying the file `filename` . If the file isn't found or can't be loaded, the resulting GtkImage will display a "broken image" icon. This function never returns `NULL`, it always returns a valid GtkImage widget.

If the file contains an animation, the image will contain an animation.

If you need to detect failures to load the file, use `gdk_pixbuf_new_from_file()` to load the file yourself, then create the GtkImage from the pixbuf. (Or for animations, use `gdk_pixbuf_animation_new_from_file()`).

The storage type (`gtk_image_get_storage_type()`) of the returned image is not defined, it will be whatever is appropriate for displaying the file.

Parameters

filename	a filename.	[type filename]

Returns

a new GtkImage

---

gtk_image_new_from_icon_set ()

```
GtkWidget *
gtk_image_new_from_icon_set (GtkIconSet *icon_set,
                             GtkIconSize size);
```

`gtk_image_new_from_icon_set` has been deprecated since version 3.10 and should not be used in newly-written code.

Use `gtk_image_new_from_icon_name()` instead.

Creates a GtkImage displaying an icon set. Sample stock sizes are GTK_ICON_SIZE_MENU, GTK_ICON_SIZE_SMALL_TOOLBAR. Instead of using this function, usually it's better to create a GtkIconFactory, put your icon sets in the icon factory, add the icon factory to the list of default factories with `gtk_icon_factory_add_default()`, and then use `gtk_image_new_from_stock()`. This will allow themes to override the icon you ship with your application.

The GtkImage does not assume a reference to the icon set; you still need to unref it if you own references. GtkImage will add its own reference rather than adopting yours.

Parameters

icon_set	a GtkIconSet	
size	a stock icon size (GtkIconSize).	[type int]

Returns

a new GtkImage

---

gtk_image_new_from_pixbuf ()

```
GtkWidget *
gtk_image_new_from_pixbuf (GdkPixbuf *pixbuf);
```

Creates a new GtkImage displaying `pixbuf` . The GtkImage does not assume a reference to the pixbuf; you still need to unref it if you own references. GtkImage will add its own reference rather than adopting yours.

Note that this function just creates an GtkImage from the pixbuf. The GtkImage created will not react to state changes. Should you want that, you should use `gtk_image_new_from_icon_name()`.

Parameters

pixbuf	a GdkPixbuf, or `NULL`.	[allow-none]

Returns

a new GtkImage

---

gtk_image_new_from_stock ()

```
GtkWidget *
gtk_image_new_from_stock (const gchar *stock_id,
                          GtkIconSize size);
```

`gtk_image_new_from_stock` has been deprecated since version 3.10 and should not be used in newly-written code.

Use `gtk_image_new_from_icon_name()` instead.

Creates a GtkImage displaying a stock icon. Sample stock icon names are GTK_STOCK_OPEN, GTK_STOCK_QUIT. Sample stock sizes are GTK_ICON_SIZE_MENU, GTK_ICON_SIZE_SMALL_TOOLBAR. If the stock icon name isn't known, the image will be empty. You can register your own stock icon names, see `gtk_icon_factory_add_default()` and `gtk_icon_factory_add()`.

Parameters

stock_id	a stock icon name	
size	a stock icon size (GtkIconSize).	[type int]

Returns

a new GtkImage displaying the stock icon

---

gtk_image_new_from_animation ()

```
GtkWidget *
gtk_image_new_from_animation (GdkPixbufAnimation *animation);
```

Creates a GtkImage displaying the given animation. The GtkImage does not assume a reference to the animation; you still need to unref it if you own references. GtkImage will add its own reference rather than adopting yours.

Note that the animation frames are shown using a timeout with G_PRIORITY_DEFAULT. When using animations to indicate busyness, keep in mind that the animation will only be shown if the main loop is not busy with something that has a higher priority.

Parameters

animation	an animation	

Returns

a new GtkImage widget

---

gtk_image_new_from_icon_name ()

```
GtkWidget *
gtk_image_new_from_icon_name (const gchar *icon_name,
                              GtkIconSize size);
```

Creates a GtkImage displaying an icon from the current icon theme. If the icon name isn't known, a "broken image" icon will be displayed instead. If the current icon theme is changed, the icon will be updated appropriately.

Parameters

icon_name	an icon name or `NULL`.	[nullable]
size	a stock icon size (GtkIconSize).	[type int]

Returns

a new GtkImage displaying the themed icon

Since: 2.6

---

gtk_image_new_from_gicon ()

```
GtkWidget *
gtk_image_new_from_gicon (GIcon *icon,
                          GtkIconSize size);
```

Creates a GtkImage displaying an icon from the current icon theme. If the icon name isn't known, a "broken image" icon will be displayed instead. If the current icon theme is changed, the icon will be updated appropriately.

Parameters

icon	an icon	
size	a stock icon size (GtkIconSize).	[type int]

Returns

a new GtkImage displaying the themed icon

Since: 2.14

---

gtk_image_new_from_resource ()

```
GtkWidget *
gtk_image_new_from_resource (const gchar *resource_path);
```

Creates a new GtkImage displaying the resource file `resource_path` . If the file isn't found or can't be loaded, the resulting GtkImage will display a "broken image" icon. This function never returns `NULL`, it always returns a valid GtkImage widget.

If the file contains an animation, the image will contain an animation.

If you need to detect failures to load the file, use `gdk_pixbuf_new_from_file()` to load the file yourself, then create the GtkImage from the pixbuf. (Or for animations, use `gdk_pixbuf_animation_new_from_file()`).

The storage type (`gtk_image_get_storage_type()`) of the returned image is not defined, it will be whatever is appropriate for displaying the file.

Parameters

resource_path	a resource path

Returns

a new GtkImage

Since: 3.4

---

gtk_image_new_from_surface ()

```
GtkWidget *
gtk_image_new_from_surface (cairo_surface_t *surface);
```

Creates a new GtkImage displaying `surface` . The GtkImage does not assume a reference to the surface; you still need to unref it if you own references. GtkImage will add its own reference rather than adopting yours.

Parameters

surface	a cairo_surface_t, or NULL.	[allow-none]

Returns

a new GtkImage

Since: 3.10

---

gtk_image_set_from_file ()

```
void
gtk_image_set_from_file (GtkImage *image,
                         const gchar *filename);
```

See gtk_image_new_from_file() for details.

Parameters

image	a GtkImage	
filename	a filename or NULL.	[type filename][allow-none]

---

gtk_image_set_from_icon_set ()

```
void
gtk_image_set_from_icon_set (GtkImage *image,
                             GtkIconSet *icon_set,
                             GtkIconSize size);
```

gtk_image_set_from_icon_set has been deprecated since version 3.10 and should not be used in newly-written code.

Use gtk_image_set_from_icon_name() instead.

See gtk_image_new_from_icon_set() for details.

Parameters

image	a GtkImage	
icon_set	a GtkIconSet	
size	a stock icon size (GtkIconSize).	[type int]

---

gtk_image_set_from_pixbuf ()

```
void
gtk_image_set_from_pixbuf (GtkImage *image,
                           GdkPixbuf *pixbuf);
```

See gtk_image_new_from_pixbuf() for details.

Parameters

image	a GtkImage	
pixbuf	a GdkPixbuf or `NULL`.	[allow-none]

---

gtk_image_set_from_stock ()

```
void
gtk_image_set_from_stock (GtkImage *image,
                          const gchar *stock_id,
                          GtkIconSize size);
```

`gtk_image_set_from_stock` has been deprecated since version 3.10 and should not be used in newly-written code.

Use `gtk_image_set_from_icon_name()` instead.

See `gtk_image_new_from_stock()` for details.

Parameters

image	a GtkImage	
stock_id	a stock icon name	
size	a stock icon size (GtkIconSize).	[type int]

---

gtk_image_set_from_animation ()

```
void
gtk_image_set_from_animation (GtkImage *image,
                              GdkPixbufAnimation *animation);
```

Causes the GtkImage to display the given animation (or display nothing, if you set the animation to `NULL`).

Parameters

image	a GtkImage
animation	the GdkPixbufAnimation

---

gtk_image_set_from_icon_name ()

```
void
gtk_image_set_from_icon_name (GtkImage *image,
                              const gchar *icon_name,
                              GtkIconSize size);
```

See `gtk_image_new_from_icon_name()` for details.

Parameters

image	a GtkImage	
icon_name	an icon name or `NULL`.	[nullable]
size	an icon size (GtkIconSize).	[type int]

Since: 2.6

---

gtk_image_set_from_gicon ()

```
void
gtk_image_set_from_gicon (GtkImage *image,
                          GIcon *icon,
                          GtkIconSize size);
```

See `gtk_image_new_from_gicon()` for details.

Parameters

image	a GtkImage	
icon	an icon	
size	an icon size (GtkIconSize).	[type int]

Since: 2.14

---

gtk_image_set_from_resource ()

```
void
gtk_image_set_from_resource (GtkImage *image,
                             const gchar *resource_path);
```

See `gtk_image_new_from_resource()` for details.

Parameters

image	a GtkImage	
resource_path	a resource path or `NULL`.	[allow-none]

---

gtk_image_set_from_surface ()

```
void
gtk_image_set_from_surface (GtkImage *image,
                            cairo_surface_t *surface);
```

See `gtk_image_new_from_surface()` for details.

Parameters

image	a GtkImage	
surface	a cairo_surface_t or `NULL`.	[nullable]

Since: 3.10

---

gtk_image_clear ()

```
void
gtk_image_clear (GtkImage *image);
```

Resets the image to be empty.

Parameters

image	a GtkImage

Since: 2.8

---

gtk_image_new ()

```
GtkWidget *
gtk_image_new (void);
```

Creates a new empty GtkImage widget.

Returns

a newly created GtkImage widget.

---

gtk_image_set_pixel_size ()

```
void
gtk_image_set_pixel_size (GtkImage *image,
                          gint pixel_size);
```

Sets the pixel size to use for named icons. If the pixel size is set to a value != -1, it is used instead of the icon size set by `gtk_image_set_from_icon_name()`.

Parameters

image	a GtkImage
pixel_size	the new pixel size

Since: 2.6

---

gtk_image_get_pixel_size ()

```
gint
gtk_image_get_pixel_size (GtkImage *image);
```

Gets the pixel size used for named icons.

Parameters

image	a GtkImage

Returns

the pixel size used for named icons.

Since: 2.6

Types and Values

struct GtkImage

```
struct GtkImage;
```

This struct contain private data only and should be accessed by the functions below.

---

enum GtkImageType

Describes the image data representation used by a GtkImage. If you want to get the image from the widget, you can only get the currently-stored representation. e.g. if the `gtk_image_get_storage_type()` returns GTK_IMAGE_PIXBUF, then you can call `gtk_image_get_pixbuf()` but not `gtk_image_get_stock()`. For empty images, you can request any storage type (call any of the "get" functions), but they will all return `NULL` values.

Members

GTK_IMAGE_EMPTY	there is no image displayed by the widget
GTK_IMAGE_PIXBUF	the widget contains a GdkPixbuf
GTK_IMAGE_STOCK	the widget contains a stock item name
GTK_IMAGE_ICON_SET	the widget contains a GtkIconSet
GTK_IMAGE_ANIMATION	the widget contains a GdkPixbufAnimation
GTK_IMAGE_ICON_NAME	the widget contains a named icon. This image type was added in GTK+ 2.6
GTK_IMAGE_GICON	the widget contains a GIcon. This image type was added in GTK+ 2.14
GTK_IMAGE_SURFACE	the widget contains a cairo_surface_t. This image type was added in GTK+ 3.10

Property Details

The “file” property

“file” gchar *

Filename to load and display.

Flags: Read / Write

Default value: NULL

The “gicon” property

“gicon” GIcon *

The GIcon displayed in the GtkImage. For themed icons, If the icon theme is changed, the image will be updated automatically.

Flags: Read / Write

Since: 2.14

The “icon-name” property

“icon-name” gchar *

The name of the icon in the icon theme. If the icon theme is changed, the image will be updated automatically.

Flags: Read / Write

Default value: NULL

Since: 2.6

The “icon-set” property

“icon-set” GtkIconSet *

Icon set to display.

`GtkImage:icon-set` has been deprecated since version 3.10 and should not be used in newly-written code.

Use “icon-name” instead.

Flags: Read / Write

The “icon-size” property

“icon-size” gint

Symbolic size to use for stock icon, icon set or named icon.

Flags: Read / Write

Allowed values: >= 0

Default value: 4

The “pixbuf” property

“pixbuf” GdkPixbuf *

A GdkPixbuf to display.

Flags: Read / Write

The “pixbuf-animation” property

“pixbuf-animation” GdkPixbufAnimation *

GdkPixbufAnimation to display.

Flags: Read / Write

The “pixel-size” property

“pixel-size” gint

The "pixel-size" property can be used to specify a fixed size overriding the “icon-size” property for images of type GTK_IMAGE_ICON_NAME.

Flags: Read / Write

Allowed values: >= -1

Default value: -1

Since: 2.6

The “resource” property

“resource” gchar *

A path to a resource file to display.

Flags: Read / Write

Default value: NULL

Since: 3.8

The “stock” property

“stock” gchar *

Stock ID for a stock image to display.

GtkImage:stock has been deprecated since version 3.10 and should not be used in newly-written code.

Use “icon-name” instead.

Flags: Read / Write

Default value: NULL

The “storage-type” property

“storage-type” GtkImageType

The representation being used for image data.

Flags: Read

Default value: GTK_IMAGE_EMPTY

The “surface” property

“surface” CairoSurface *

A cairo_surface_t to display.

Flags: Read / Write

The “use-fallback” property

“use-fallback” gboolean

Whether the icon displayed in the GtkImage will use standard icon names fallback. The value of this property is only relevant for images of type GTK_IMAGE_ICON_NAME and GTK_IMAGE_GICON.

Flags: Read / Write

Default value: FALSE

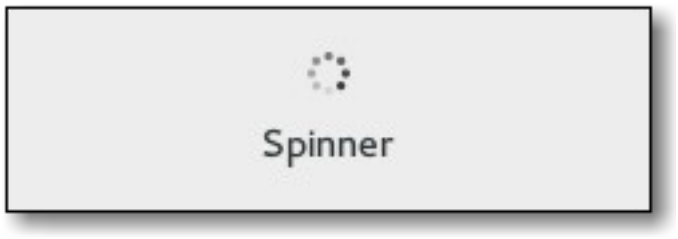
Since: 3.0

See Also

GdkPixbuf

GtkSpinner

GtkSpinner — Show a spinner animation



Functions

GtkWidget *	gtk_spinner_new ()
void	gtk_spinner_start ()
void	gtk_spinner_stop ()

Properties

gboolean	active	Read / Write

Types and Values

struct	GtkSpinner

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkSpinner
```

Implemented Interfaces

GtkSpinner implements AtkImplementorIface and GtkBuildable.

Includes

```
#include <gtk/gtk.h>
```

Description

A GtkSpinner widget displays an icon-size spinning animation. It is often used as an alternative to a GtkProgressBar for displaying indefinite activity, instead of actual progress.

To start the animation, use `gtk_spinner_start()`, to stop it use `gtk_spinner_stop()`.

CSS nodes

GtkSpinner has a single CSS node with the name spinner. When the animation is active, the :checked pseudoclass is added to this node.

Functions

gtk_spinner_new ()

```
GtkWidget *
gtk_spinner_new (void);
```

Returns a new spinner widget. Not yet started.

Returns

a new GtkSpinner

Since: 2.20

---

gtk_spinner_start ()

```
void
gtk_spinner_start (GtkSpinner *spinner);
```

Starts the animation of the spinner.

Parameters

spinner	a GtkSpinner

Since: 2.20

---

gtk_spinner_stop ()

```
void
gtk_spinner_stop (GtkSpinner *spinner);
```

Stops the animation of the spinner.

Parameters

spinner	a GtkSpinner

Since: 2.20

Types and Values

struct GtkSpinner

```
struct GtkSpinner;
```

Property Details

The "active" property

```
  "active"                   gboolean
```

Whether the spinner is active.

Flags: Read / Write

Default value: FALSE

See Also

GtkCellRendererSpinner, GtkProgressBar

GtkInfoBar

GtkInfoBar — Report important messages to the user

Functions

GtkWidget *	gtk_info_bar_new ()
GtkWidget *	gtk_info_bar_new_with_buttons ()
void	gtk_info_bar_add_action_widget ()
GtkWidget *	gtk_info_bar_add_button ()
void	gtk_info_bar_add_buttons ()
void	gtk_info_bar_set_response_sensitive ()
void	gtk_info_bar_set_default_response ()
void	gtk_info_bar_response ()
void	gtk_info_bar_set_message_type ()
GtkMessageType	gtk_info_bar_get_message_type ()
GtkWidget *	gtk_info_bar_get_action_area ()
GtkWidget *	gtk_info_bar_get_content_area ()
gboolean	gtk_info_bar_get_show_close_button ()

void	gtk_info_bar_set_show_close_button ()
gboolean	gtk_info_bar_get_revealed ()
void	gtk_info_bar_set_revealed ()

Properties

GtkMessageType	message-type	Read / Write / Construct
gboolean	revealed	Read / Write
gboolean	show-close-button	Read / Write / Construct

Style Properties

gint	action-area-border	Read
gint	button-spacing	Read
gint	content-area-border	Read
gint	content-area-spacing	Read

Signals

void	close	Action
void	response	Run Last

Types and Values

struct	GtkInfoBar

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkContainer
            └── GtkBox
                └── GtkInfoBar
```

Implemented Interfaces

GtkInfoBar implements AtkImplementorIface, GtkBuildable and GtkOrientable.

Includes

```
#include <gtk/gtk.h>
```

Description

GtkInfoBar is a widget that can be used to show messages to the user without showing a dialog. It is often temporarily shown at the top or bottom of a document. In contrast to GtkDialog, which has a action area at the bottom, GtkInfoBar has an action area at the side.

The API of GtkInfoBar is very similar to GtkDialog, allowing you to add buttons to the action area with `gtk_info_bar_add_button()` or `gtk_info_bar_new_with_buttons()`. The sensitivity of action widgets can be controlled with `gtk_info_bar_set_response_sensitive()`. To add widgets to the main content area of a GtkInfoBar, use `gtk_info_bar_get_content_area()` and add your widgets to the container.

Similar to GtkMessageDialog, the contents of a GtkInfoBar can by classified as error message, warning, informational message, etc, by using `gtk_info_bar_set_message_type()`. GTK+ may use the message type to determine how the message is displayed.

A simple example for using a GtkInfoBar:

```
1  GtkWidget *widget, *message_label, *content_area;
2  GtkWidget *grid;
3  GtkInfoBar *bar;
4
5  // set up info bar
6  widget = gtk_info_bar_new ();
7  bar = GTK_INFO_BAR (widget);
8  grid = gtk_grid_new ();
9
10 gtk_widget_set_no_show_all (widget, TRUE);
11 message_label = gtk_label_new ("");
12 content_area = gtk_info_bar_get_content_area (bar);
13 gtk_container_add (GTK_CONTAINER (content_area),
14                     message_label);
15 gtk_info_bar_add_button (bar,
16                          _("_OK"),
17                          GTK_RESPONSE_OK);
18 g_signal_connect (bar,
19                   "response",
20                   G_CALLBACK (gtk_widget_hide),
21                   NULL);
22 gtk_grid_attach (GTK_GRID (grid),
23                  widget,
24                  0, 2, 1, 1);
25
26 // ...
27
28 // show an error message
29 gtk_label_set_text (GTK_LABEL (message_label), "An error occurred!");
30 gtk_info_bar_set_message_type (bar,
31                                GTK_MESSAGE_ERROR);
32 gtk_widget_show (bar);
```

GtkInfoBar as GtkBuildable

The GtkInfoBar implementation of the GtkBuildable interface exposes the content area and action area as internal children with the names “content_area” and “action_area”.

GtkInfoBar supports a custom <action-widgets> element, which can contain multiple <action-widget> elements. The “response” attribute specifies a numeric response, and the content of the element is the id of widget (which should be a child of the dialogs `action_area`).

CSS nodes

GtkInfoBar has a single CSS node with name infobar. The node may get one of the style classes .info, .warning, .error or .question, depending on the message type.

Functions

gtk_info_bar_new ()

```
GtkWidget *
gtk_info_bar_new (void);
```

Creates a new GtkInfoBar object.

Returns

a new GtkInfoBar object

Since: 2.18

---

gtk_info_bar_new_with_buttons ()

```
GtkWidget *
gtk_info_bar_new_with_buttons (const gchar *first_button_text,
                               ...);
```

Creates a new GtkInfoBar with buttons. Button text/response ID pairs should be listed, with a `NULL` pointer ending the list. Button text can be either a stock ID such as `GTK_STOCK_OK`, or some arbitrary text. A response ID can be any positive number, or one of the values in the GtkResponseType enumeration. If the user clicks one of these dialog buttons, GtkInfoBar will emit the “response” signal with the corresponding response ID.

Parameters

first_button_text	stock ID or text to go in first button, or `NULL`.	[allow-none]
...	response ID for first button, then additional buttons, ending with `NULL`	

Returns

a new GtkInfoBar

---

gtk_info_bar_add_action_widget ()

```
void
gtk_info_bar_add_action_widget (GtkInfoBar *info_bar,
```

```
                                GtkWidget *child,
                                gint response_id);
```

Add an activatable widget to the action area of a GtkInfoBar, connecting a signal handler that will emit the "response" signal on the message area when the widget is activated. The widget is appended to the end of the message areas action area.

Parameters

info_bar	a GtkInfoBar
child	an activatable widget
response_id	response ID for `child`

Since: 2.18

---

gtk_info_bar_add_button ()

```
GtkWidget *
gtk_info_bar_add_button (GtkInfoBar *info_bar,
                         const gchar *button_text,
                         gint response_id);
```

Adds a button with the given text and sets things up so that clicking the button will emit the "response" signal with the given response_id. The button is appended to the end of the info bars's action area. The button widget is returned, but usually you don't need it.

Parameters

info_bar	a GtkInfoBar
button_text	text of button
response_id	response ID for the button

Returns

the GtkButton widget that was added.

[transfer none][type Gtk.Button]

Since: 2.18

---

gtk_info_bar_add_buttons ()

```
void
gtk_info_bar_add_buttons (GtkInfoBar *info_bar,
                          const gchar *first_button_text,
                          ...);
```

Adds more buttons, same as calling `gtk_info_bar_add_button()` repeatedly. The variable argument list should be `NULL`-terminated as with `gtk_info_bar_new_with_buttons()`. Each button must have both text and response ID.

Parameters

info_bar	a GtkInfoBar
first_button_text	button text or stock ID
...	response ID for first button, then more text-response_id pairs, ending with `NULL`

Since: 2.18

---

gtk_info_bar_set_response_sensitive ()

```
void
gtk_info_bar_set_response_sensitive (GtkInfoBar *info_bar,
                                     gint response_id,
                                     gboolean setting);
```

Calls gtk_widget_set_sensitive (widget, setting) for each widget in the info bars's action area with the given response_id. A convenient way to sensitize/desensitize dialog buttons.

Parameters

info_bar	a GtkInfoBar
response_id	a response ID
setting	TRUE for sensitive

Since: 2.18

---

gtk_info_bar_set_default_response ()

```
void
gtk_info_bar_set_default_response (GtkInfoBar *info_bar,
                                   gint response_id);
```

Sets the last widget in the info bar's action area with the given response_id as the default widget for the dialog. Pressing "Enter" normally activates the default widget.

Note that this function currently requires `info_bar` to be added to a widget hierarchy.

Parameters

info_bar	a GtkInfoBar
response_id	a response ID

Since: 2.18

---

gtk_info_bar_response ()

```
void
gtk_info_bar_response (GtkInfoBar *info_bar,
                       gint response_id);
```

Emits the “response” signal with the given `response_id`.

Parameters

info_bar	a GtkInfoBar
response_id	a response ID

Since: 2.18

---

gtk_info_bar_set_message_type ()

```
void
gtk_info_bar_set_message_type (GtkInfoBar *info_bar,
                               GtkMessageType message_type);
```

Sets the message type of the message area.

GTK+ uses this type to determine how the message is displayed.

Parameters

info_bar	a GtkInfoBar
message_type	a GtkMessageType

Since: 2.18

---

gtk_info_bar_get_message_type ()

```
GtkMessageType
gtk_info_bar_get_message_type (GtkInfoBar *info_bar);
```

Returns the message type of the message area.

Parameters

info_bar	a GtkInfoBar

Returns

the message type of the message area.

Since: 2.18

---

gtk_info_bar_get_action_area ()

```
GtkWidget *
gtk_info_bar_get_action_area (GtkInfoBar *info_bar);
```

Returns the action area of `info_bar`.

Parameters

info_bar	a GtkInfoBar

Returns

the action area.

[transfer none]

Since: 2.18

---

gtk_info_bar_get_content_area ()

```
GtkWidget *
gtk_info_bar_get_content_area (GtkInfoBar *info_bar);
```

Returns the content area of `info_bar` .

Parameters

info_bar	a GtkInfoBar

Returns

the content area.

[transfer none]

Since: 2.18

---

gtk_info_bar_get_show_close_button ()

```
gboolean
gtk_info_bar_get_show_close_button (GtkInfoBar *info_bar);
```

Returns whether the widget will display a standard close button.

Parameters

info_bar	a GtkInfoBar

Returns

`TRUE` if the widget displays standard close button

Since: 3.10

---

gtk_info_bar_set_show_close_button ()

```
void
gtk_info_bar_set_show_close_button (GtkInfoBar *info_bar,
                                    gboolean setting);
```

If true, a standard close button is shown. When clicked it emits the response `GTK_RESPONSE_CLOSE`.

Parameters

info_bar	a GtkInfoBar
setting	`TRUE` to include a close button

Since: 3.10

---

gtk_info_bar_get_revealed ()

```
gboolean
gtk_info_bar_get_revealed (GtkInfoBar *info_bar);
```

Parameters

info_bar	a GtkInfoBar

Returns

the current value of the GtkInfoBar:revealed property.

Since: 3.22.29

---

gtk_info_bar_set_revealed ()

```
void
gtk_info_bar_set_revealed (GtkInfoBar *info_bar,
                           gboolean revealed);
```

Sets the GtkInfoBar:revealed property to `revealed` . This will cause `info_bar` to show up with a slide-in transition.

Note that this property does not automatically show `info_bar` and thus won't have any effect if it is invisible.

Parameters

info_bar	a GtkInfoBar
revealed	The new value of the property

Since: 3.22.29

Types and Values

struct GtkInfoBar

```
struct GtkInfoBar;
```

Property Details

The “message-type” property

```
  “message-type”              GtkMessageType
```

The type of the message.

The type may be used to determine the appearance of the info bar.

Flags: Read / Write / Construct

Default value: GTK_MESSAGE_INFO

Since: 2.18

The “revealed” property

```
  “revealed”                  gboolean
```

Controls whether the action bar shows its contents or not.

Flags: Read / Write

Default value: TRUE

The “show-close-button” property

```
  “show-close-button”         gboolean
```

Whether to include a standard close button.

Flags: Read / Write / Construct

Default value: FALSE

Since: 3.10

Style Property Details

The “action-area-border” style property

```
  “action-area-border”        gint
```

Width of the border around the action area of the info bar.

`GtkInfoBar:action-area-border` has been deprecated since version 3.6 and should not be used in newly-written code.

Use `gtk_container_set_border_width()`

Flags: Read

Allowed values: >= 0

Default value: 5

Since: 2.18

The “button-spacing” style property

`“button-spacing”` `gint`

Spacing between buttons in the action area of the info bar.

`GtkInfoBar:button-spacing` has been deprecated since version 3.6 and should not be used in newly-written code.

Use `gtk_box_set_spacing()`

Flags: Read

Allowed values: >= 0

Default value: 6

Since: 2.18

The “content-area-border” style property

`“content-area-border”` `gint`

The width of the border around the content content area of the info bar.

`GtkInfoBar:content-area-border` has been deprecated since version 3.6 and should not be used in newly-written code.

Use `gtk_container_set_border_width()`

Flags: Read

Allowed values: >= 0

Default value: 8

Since: 2.18

The “content-area-spacing” style property

```
  “content-area-spacing”     gint
```

The default spacing used between elements of the content area of the info bar.

`GtkInfoBar:content-area-spacing` has been deprecated since version 3.6 and should not be used in newly-written code.

Use `gtk_box_set_spacing()`

Flags: Read

Allowed values: >= 0

Default value: 16

Since: 2.18

Signal Details

The “close” signal

```
void
user_function (GtkInfoBar *infobar,
               gpointer    user_data)
```

The ::close signal is a keybinding signal which gets emitted when the user uses a keybinding to dismiss the info bar.

The default binding for this signal is the Escape key.

Parameters

user_data	user data set when the signal handler was connected.

Flags: Action

Since: 2.18

The “response” signal

```
void
user_function (GtkInfoBar *info_bar,
               gint        response_id,
               gpointer    user_data)
```

Emitted when an action widget is clicked or the application programmer calls `gtk_dialog_response()`. The `response_id` depends on which action widget was clicked.

Parameters

info_bar	the object on which the signal is

	emitted
response_id	the response ID
user_data	user data set when the signal handler was connected.

Flags: Run Last

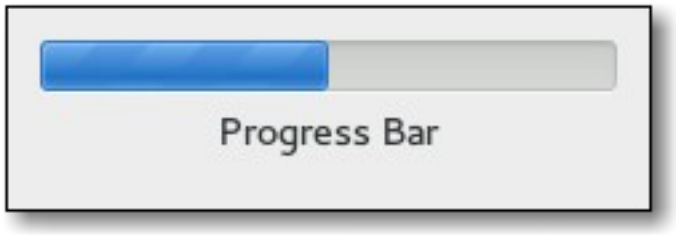
Since: 2.18

See Also

GtkStatusbar, GtkMessageDialog

GtkProgressBar

GtkProgressBar — A widget which indicates progress visually



Functions

GtkWidget *	gtk_progress_bar_new ()
void	gtk_progress_bar_pulse ()
void	gtk_progress_bar_set_fraction ()
gdouble	gtk_progress_bar_get_fraction ()
void	gtk_progress_bar_set_inverted ()
gboolean	gtk_progress_bar_get_inverted ()
void	gtk_progress_bar_set_show_text ()
gboolean	gtk_progress_bar_get_show_text ()
void	gtk_progress_bar_set_text ()
const gchar *	gtk_progress_bar_get_text ()
void	gtk_progress_bar_set_ellipsize ()
PangoEllipsizeMode	gtk_progress_bar_get_ellipsize ()
void	gtk_progress_bar_set_pulse_step ()
gdouble	gtk_progress_bar_get_pulse_step ()

Properties

PangoEllipsizeMode	ellipsize	Read / Write
gdouble	fraction	Read / Write
gboolean	inverted	Read / Write
gdouble	pulse-step	Read / Write
gboolean	show-text	Read / Write
gchar *	text	Read / Write

Style Properties

gint	min-horizontal-bar-height	Read / Write
gint	min-horizontal-bar-width	Read / Write
gint	min-vertical-bar-height	Read / Write
gint	min-vertical-bar-width	Read / Write
gint	xspacing	Read / Write
gint	yspacing	Read / Write

Types and Values

struct	GtkProgressBar

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkProgressBar
```

Implemented Interfaces

GtkProgressBar implements AtkImplementorIface, GtkBuildable and GtkOrientable.

Includes

```
#include <gtk/gtk.h>
```

Description

The GtkProgressBar is typically used to display the progress of a long running operation. It provides a visual clue that processing is underway. The GtkProgressBar can be used in two different modes: percentage mode and activity mode.

When an application can determine how much work needs to take place (e.g. read a fixed number of bytes from a file) and can monitor its progress, it can use the GtkProgressBar in percentage mode and the user sees a growing bar indicating the percentage of the work that has been completed. In this mode, the application is required to call `gtk_progress_bar_set_fraction()` periodically to update the progress bar.

When an application has no accurate way of knowing the amount of work to do, it can use the GtkProgressBar in activity mode, which shows activity by a block moving back and forth within the progress area. In this mode, the application is required to call `gtk_progress_bar_pulse()` periodically to update the progress bar.

There is quite a bit of flexibility provided to control the appearance of the GtkProgressBar. Functions are provided to control the orientation of the bar, optional text can be displayed along with the bar, and the step size used in activity mode can be set.

CSS nodes

```
progressbar[.osd]
├── [text]
└── trough[.empty][.full]
    └── progress[.pulse]
```

GtkProgressBar has a main CSS node with name progressbar and subnodes with names text and trough, of which the latter has a subnode named progress. The text subnode is only present if text is shown. The progress subnode has the style class .pulse when in activity mode. It gets the style classes .left, .right, .top or .bottom added when the progress 'touches' the corresponding end of the GtkProgressBar. The .osd class on the progressbar node is for use in overlays like the one Epiphany has for page loading progress.

Functions

gtk_progress_bar_new ()

```
GtkWidget *
gtk_progress_bar_new (void);
```

Creates a new GtkProgressBar.

Returns

a GtkProgressBar.

---

gtk_progress_bar_pulse ()

```
void
gtk_progress_bar_pulse (GtkProgressBar *pbar);
```

Indicates that some progress has been made, but you don’t know how much. Causes the progress bar to enter “activity mode,” where a block bounces back and forth. Each call to `gtk_progress_bar_pulse()` causes the block to move by a little bit (the amount of movement per pulse is determined by `gtk_progress_bar_set_pulse_step()`).

Parameters

pbar	a GtkProgressBar

---

gtk_progress_bar_set_fraction ()

```
void
gtk_progress_bar_set_fraction (GtkProgressBar *pbar,
                               gdouble fraction);
```

Causes the progress bar to “fill in” the given fraction of the bar. The fraction should be between 0.0 and 1.0, inclusive.

Parameters

pbar	a GtkProgressBar
fraction	fraction of the task that's been completed

---

gtk_progress_bar_get_fraction ()

```
gdouble
gtk_progress_bar_get_fraction (GtkProgressBar *pbar);
```

Returns the current fraction of the task that's been completed.

Parameters

pbar	a GtkProgressBar

Returns

a fraction from 0.0 to 1.0

---

gtk_progress_bar_set_inverted ()

```
void
gtk_progress_bar_set_inverted (GtkProgressBar *pbar,
                               gboolean inverted);
```

Progress bars normally grow from top to bottom or left to right. Inverted progress bars grow in the opposite direction.

Parameters

pbar	a GtkProgressBar
inverted	`TRUE` to invert the progress bar

---

gtk_progress_bar_get_inverted ()

```
gboolean
gtk_progress_bar_get_inverted (GtkProgressBar *pbar);
```

Gets the value set by `gtk_progress_bar_set_inverted()`.

Parameters

pbar	a GtkProgressBar

Returns

TRUE if the progress bar is inverted

---

gtk_progress_bar_set_show_text ()

```
void
gtk_progress_bar_set_show_text (GtkProgressBar *pbar,
                                gboolean show_text);
```

Sets whether the progress bar will show text next to the bar. The shown text is either the value of the "text" property or, if that is NULL, the "fraction" value, as a percentage.

To make a progress bar that is styled and sized suitably for containing text (even if the actual text is blank), set "show-text" to TRUE and "text" to the empty string (not NULL).

Parameters

pbar	a GtkProgressBar
show_text	whether to show text

Since: 3.0

---

gtk_progress_bar_get_show_text ()

```
gboolean
gtk_progress_bar_get_show_text (GtkProgressBar *pbar);
```

Gets the value of the "show-text" property. See `gtk_progress_bar_set_show_text()`.

Parameters

pbar	a GtkProgressBar

Returns

TRUE if text is shown in the progress bar

Since: 3.0

---

gtk_progress_bar_set_text ()

```
void
gtk_progress_bar_set_text (GtkProgressBar *pbar,
                           const gchar *text);
```

Causes the given text to appear next to the progress bar.

If text is NULL and "show-text" is TRUE, the current value of "fraction" will be displayed as a percentage.

If text is non-NULL and "show-text" is TRUE, the text will be displayed. In this case, it will not display the

progress percentage. If `text` is the empty string, the progress bar will still be styled and sized suitably for containing text, as long as “show-text” is `TRUE`.

Parameters

pbar	a GtkProgressBar	
text	a UTF-8 string, or `NULL`.	[allow-none]

---

gtk_progress_bar_get_text ()

```
const gchar *
gtk_progress_bar_get_text (GtkProgressBar *pbar);
```

Retrieves the text that is displayed with the progress bar, if any, otherwise `NULL`. The return value is a reference to the text, not a copy of it, so will become invalid if you change the text in the progress bar.

Parameters

pbar	a GtkProgressBar

Returns

text, or `NULL`; this string is owned by the widget and should not be modified or freed.

[nullable]

---

gtk_progress_bar_set_ellipsize ()

```
void
gtk_progress_bar_set_ellipsize (GtkProgressBar *pbar,
                                PangoEllipsizeMode mode);
```

Sets the mode used to ellipsize (add an ellipsis: "...") the text if there is not enough space to render the entire string.

Parameters

pbar	a GtkProgressBar
mode	a PangoEllipsizeMode

Since: 2.6

---

gtk_progress_bar_get_ellipsize ()

```
PangoEllipsizeMode
gtk_progress_bar_get_ellipsize (GtkProgressBar *pbar);
```

Returns the ellipsizing position of the progress bar. See `gtk_progress_bar_set_ellipsize()`.

Parameters

pbar	a GtkProgressBar

Returns

PangoEllipsizeMode

Since: 2.6

---

gtk_progress_bar_set_pulse_step ()

```
void
gtk_progress_bar_set_pulse_step (GtkProgressBar *pbar,
                                 gdouble fraction);
```

Sets the fraction of total progress bar length to move the bouncing block for each call to `gtk_progress_bar_pulse()`.

Parameters

pbar	a GtkProgressBar
fraction	fraction between 0.0 and 1.0

---

gtk_progress_bar_get_pulse_step ()

```
gdouble
gtk_progress_bar_get_pulse_step (GtkProgressBar *pbar);
```

Retrieves the pulse step set with `gtk_progress_bar_set_pulse_step()`.

Parameters

pbar	a GtkProgressBar

Returns

a fraction from 0.0 to 1.0

Types and Values

struct GtkProgressBar

```
struct GtkProgressBar;
```

Property Details

The `“ellipsize”` property

`“ellipsize”` `PangoEllipsizeMode`

The preferred place to ellipsize the string, if the progress bar does not have enough room to display the entire string, specified as a PangoEllipsizeMode.

Note that setting this property to a value other than `PANGO_ELLIPSIZE_NONE` has the side-effect that the progress bar requests only enough space to display the ellipsis ("..."). Another means to set a progress bar's width is `gtk_widget_set_size_request()`.

Flags: Read / Write

Default value: PANGO_ELLIPSIZE_NONE

Since: 2.6

The `“fraction”` property

`“fraction”` `gdouble`

The fraction of total work that has been completed.

Flags: Read / Write

Allowed values: [0,1]

Default value: 0

The `“inverted”` property

`“inverted”` `gboolean`

Invert the direction in which the progress bar grows.

Flags: Read / Write

Default value: FALSE

The `“pulse-step”` property

`“pulse-step”` `gdouble`

The fraction of total progress to move the bouncing block when pulsed.

Flags: Read / Write

Allowed values: [0,1]

Default value: 0.1

The “show-text” property

“show-text” gboolean

Sets whether the progress bar will show a text in addition to the bar itself. The shown text is either the value of the “text” property or, if that is NULL, the “fraction” value, as a percentage.

To make a progress bar that is styled and sized suitably for showing text (even if the actual text is blank), set “show-text” to TRUE and “text” to the empty string (not NULL).

Flags: Read / Write

Default value: FALSE

Since: 3.0

The “text” property

“text” gchar *

Text to be displayed in the progress bar.

Flags: Read / Write

Default value: NULL

Style Property Details

The “min-horizontal-bar-height” style property

“min-horizontal-bar-height” gint

Minimum horizontal height of the progress bar.

GtkProgressBar:min-horizontal-bar-height has been deprecated since version 3.20 and should not be used in newly-written code.

Use the standard CSS property min-height.

Flags: Read / Write

Allowed values: >= 1

Default value: 6

Since: 2.14

The “min-horizontal-bar-width” style property

“min-horizontal-bar-width” gint

The minimum horizontal width of the progress bar.

`GtkProgressBar:min-horizontal-bar-width` has been deprecated since version 3.20 and should not be used in newly-written code.

Use the standard CSS property min-width.

Flags: Read / Write

Allowed values: >= 1

Default value: 150

Since: 2.14

The `"min-vertical-bar-height"` style property

`"min-vertical-bar-height" gint`

The minimum vertical height of the progress bar.

`GtkProgressBar:min-vertical-bar-height` has been deprecated since version 3.20 and should not be used in newly-written code.

Use the standard CSS property min-height.

Flags: Read / Write

Allowed values: >= 1

Default value: 80

Since: 2.14

The `"min-vertical-bar-width"` style property

`"min-vertical-bar-width" gint`

The minimum vertical width of the progress bar.

`GtkProgressBar:min-vertical-bar-width` has been deprecated since version 3.20 and should not be used in newly-written code.

Use the standard CSS property min-width.

Flags: Read / Write

Allowed values: >= 1

Default value: 7

Since: 2.14

The “xspacing” style property

“xspacing” gint

Extra spacing applied to the width of a progress bar.

GtkProgressBar:xspacing has been deprecated since version 3.20 and should not be used in newly-written code.

Use the standard CSS padding and margins; the value of this style property is ignored.

Flags: Read / Write

Allowed values: >= 0

Default value: 2

The “yspacing” style property

“yspacing” gint

Extra spacing applied to the height of a progress bar.

GtkProgressBar:yspacing has been deprecated since version 3.20 and should not be used in newly-written code.

Use the standard CSS padding and margins; the value of this style property is ignored.

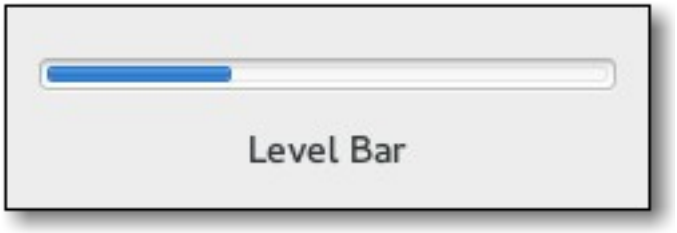
Flags: Read / Write

Allowed values: >= 0

Default value: 2

GtkLevelBar

GtkLevelBar — A bar that can used as a level indicator



Functions

GtkWidget *	gtk_level_bar_new ()
GtkWidget *	gtk_level_bar_new_for_interval ()
void	gtk_level_bar_set_mode ()
GtkLevelBarMode	gtk_level_bar_get_mode ()
void	gtk_level_bar_set_value ()
gdouble	gtk_level_bar_get_value ()
void	gtk_level_bar_set_min_value ()
gdouble	gtk_level_bar_get_min_value ()
void	gtk_level_bar_set_max_value ()
gdouble	gtk_level_bar_get_max_value ()
void	gtk_level_bar_set_inverted ()
gboolean	gtk_level_bar_get_inverted ()
void	gtk_level_bar_add_offset_value ()
void	gtk_level_bar_remove_offset_value ()
gboolean	gtk_level_bar_get_offset_value ()

Properties

gboolean	inverted	Read / Write
gdouble	max-value	Read / Write
gdouble	min-value	Read / Write
GtkLevelBarMode	mode	Read / Write
gdouble	value	Read / Write

Style Properties

gint	min-block-height	Read / Write
gint	min-block-width	Read / Write

Signals

void	offset-changed	Has Details

Types and Values

#define	GTK_LEVEL_BAR_OFFSET_LOW

#define	GTK_LEVEL_BAR_OFFSET_HIGH
#define	GTK_LEVEL_BAR_OFFSET_FULL
enum	GtkLevelBarMode
struct	GtkLevelBar

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkLevelBar
```

Implemented Interfaces

GtkLevelBar implements AtkImplementorIface, GtkBuildable and GtkOrientable.

Includes

```
#include <gtk/gtk.h>
```

Description

The GtkLevelBar is a bar widget that can be used as a level indicator. Typical use cases are displaying the strength of a password, or showing the charge level of a battery.

Use `gtk_level_bar_set_value()` to set the current value, and `gtk_level_bar_add_offset_value()` to set the value offsets at which the bar will be considered in a different state. GTK will add a few offsets by default on the level bar: GTK_LEVEL_BAR_OFFSET_LOW, GTK_LEVEL_BAR_OFFSET_HIGH and GTK_LEVEL_BAR_OFFSET_FULL, with values 0.25, 0.75 and 1.0 respectively.

Note that it is your responsibility to update preexisting offsets when changing the minimum or maximum value. GTK+ will simply clamp them to the new range.

Adding a custom offset on the bar

```
1   static GtkWidget *
2   create_level_bar (void)
3   {
4     GtkWidget *widget;
5     GtkLevelBar *bar;
6
7     widget = gtk_level_bar_new ();
8     bar = GTK_LEVEL_BAR (widget);
9
10    // This changes the value of the default low offset
11
12    gtk_level_bar_add_offset_value (bar,
13                                    GTK_LEVEL_BAR_OFFSET_LOW,
14                                    0.10);
15
16    // This adds a new offset to the bar; the application will
17    // be able to change its color CSS like this:
18    //
```

```
19    // levelbar block.my-offset {
20    //   background-color: magenta;
21    //   border-style: solid;
22    //   border-color: black;
23    //   border-style: 1px;
24    // }
25
26    gtk_level_bar_add_offset_value (bar, "my-offset", 0.60);
27
28    return widget;
29  }
```

The default interval of values is between zero and one, but it’s possible to modify the interval using `gtk_level_bar_set_min_value()` and `gtk_level_bar_set_max_value()`. The value will be always drawn in proportion to the admissible interval, i.e. a value of 15 with a specified interval between 10 and 20 is equivalent to a value of 0.5 with an interval between 0 and 1. When GTK_LEVEL_BAR_MODE_DISCRETE is used, the bar level is rendered as a finite number of separated blocks instead of a single one. The number of blocks that will be rendered is equal to the number of units specified by the admissible interval.

For instance, to build a bar rendered with five blocks, it’s sufficient to set the minimum value to 0 and the maximum value to 5 after changing the indicator mode to discrete.

GtkLevelBar was introduced in GTK+ 3.6.

GtkLevelBar as GtkBuildable

The GtkLevelBar implementation of the GtkBuildable interface supports a custom <offsets> element, which can contain any number of <offset> elements, each of which must have name and value attributes.

CSS nodes

```
1   levelbar[.discrete]
2   └── trough
3       ├── block.filled.level-name
4       ┊
5       ├── block.empty
6       ┊
```

GtkLevelBar has a main CSS node with name levelbar and one of the style classes .discrete or .continuous and a subnode with name trough. Below the trough node are a number of nodes with name block and style class .filled or .empty. In continuous mode, there is exactly one node of each, in discrete mode, the number of filled and unfilled nodes corresponds to blocks that are drawn. The block.filled nodes also get a style class .level-name corresponding to the level for the current value.

In horizontal orientation, the nodes are always arranged from left to right, regardless of text direction.

Functions

gtk_level_bar_new ()

```
GtkWidget *
gtk_level_bar_new (void);
```

Creates a new GtkLevelBar.

Returns

a GtkLevelBar.

Since: 3.6

---

gtk_level_bar_new_for_interval ()

```
GtkWidget *
gtk_level_bar_new_for_interval (gdouble min_value,
                                gdouble max_value);
```

Utility constructor that creates a new GtkLevelBar for the specified interval.

Parameters

min_value	a positive value
max_value	a positive value

Returns

a GtkLevelBar

Since: 3.6

---

gtk_level_bar_set_mode ()

```
void
gtk_level_bar_set_mode (GtkLevelBar *self,
                        GtkLevelBarMode mode);
```

Sets the value of the “mode” property.

Parameters

self	a GtkLevelBar
mode	a GtkLevelBarMode

Since: 3.6

---

gtk_level_bar_get_mode ()

```
GtkLevelBarMode
gtk_level_bar_get_mode (GtkLevelBar *self);
```

Returns the value of the “mode” property.

Parameters

self	a GtkLevelBar

Returns

a GtkLevelBarMode

Since: 3.6

---

gtk_level_bar_set_value ()

```
void
gtk_level_bar_set_value (GtkLevelBar *self,
                         gdouble value);
```

Sets the value of the “value” property.

Parameters

self	a GtkLevelBar
value	a value in the interval between “min-value” and “max-value”

Since: 3.6

---

gtk_level_bar_get_value ()

```
gdouble
gtk_level_bar_get_value (GtkLevelBar *self);
```

Returns the value of the “value” property.

Parameters

self	a GtkLevelBar

Returns

a value in the interval between “min-value” and “max-value”

Since: 3.6

---

gtk_level_bar_set_min_value ()

```
void
```

```
gtk_level_bar_set_min_value (GtkLevelBar *self,
                             gdouble value);
```

Sets the value of the “min-value” property.

You probably want to update preexisting level offsets after calling this function.

Parameters

self	a GtkLevelBar
value	a positive value

Since: 3.6

---

gtk_level_bar_get_min_value ()

```
gdouble
gtk_level_bar_get_min_value (GtkLevelBar *self);
```

Returns the value of the “min-value” property.

Parameters

self	a GtkLevelBar

Returns

a positive value

Since: 3.6

---

gtk_level_bar_set_max_value ()

```
void
gtk_level_bar_set_max_value (GtkLevelBar *self,
                             gdouble value);
```

Sets the value of the “max-value” property.

You probably want to update preexisting level offsets after calling this function.

Parameters

self	a GtkLevelBar
value	a positive value

Since: 3.6

---

gtk_level_bar_get_max_value ()

```
gdouble
gtk_level_bar_get_max_value (GtkLevelBar *self);
```

Returns the value of the “max-value” property.

Parameters

self	a GtkLevelBar

Returns

a positive value

Since: 3.6

---

gtk_level_bar_set_inverted ()

```
void
gtk_level_bar_set_inverted (GtkLevelBar *self,
                            gboolean inverted);
```

Sets the value of the “inverted” property.

Parameters

self	a GtkLevelBar
inverted	TRUE to invert the level bar

Since: 3.8

---

gtk_level_bar_get_inverted ()

```
gboolean
gtk_level_bar_get_inverted (GtkLevelBar *self);
```

Return the value of the “inverted” property.

Parameters

self	a GtkLevelBar

Returns

TRUE if the level bar is inverted

Since: 3.8

---

gtk_level_bar_add_offset_value ()

```
void
gtk_level_bar_add_offset_value (GtkLevelBar *self,
                                const gchar *name,
                                gdouble value);
```

Adds a new offset marker on `self` at the position specified by `value` . When the bar value is in the interval topped by `value` (or between `value` and "max-value" in case the offset is the last one on the bar) a style class named `level-name` will be applied when rendering the level bar fill. If another offset marker named `name` exists, its value will be replaced by `value` .

Parameters

self	a GtkLevelBar	
name	the name of the new offset	
value	the value for the new offset	

Since: 3.6

---

gtk_level_bar_remove_offset_value ()

```
void
gtk_level_bar_remove_offset_value (GtkLevelBar *self,
                                   const gchar *name);
```

Removes an offset marker previously added with `gtk_level_bar_add_offset_value()`.

Parameters

self	a GtkLevelBar	
name	the name of an offset in the bar.	[allow-none]

Since: 3.6

---

gtk_level_bar_get_offset_value ()

```
gboolean
gtk_level_bar_get_offset_value (GtkLevelBar *self,
                                const gchar *name,
                                gdouble *value);
```

Fetches the value specified for the offset marker `name` in `self` , returning `TRUE` in case an offset named `name` was found.

Parameters

self	a GtkLevelBar	
name	the name of an offset in the bar.	[allow-none]
value	location where to store the value.	[out]

Returns

`TRUE` if the specified offset is found

Since: 3.6

Types and Values

GTK_LEVEL_BAR_OFFSET_LOW

```
#define GTK_LEVEL_BAR_OFFSET_LOW  "low"
```

The name used for the stock low offset included by GtkLevelBar.

Since: 3.6

GTK_LEVEL_BAR_OFFSET_HIGH

```
#define GTK_LEVEL_BAR_OFFSET_HIGH "high"
```

The name used for the stock high offset included by GtkLevelBar.

Since: 3.6

GTK_LEVEL_BAR_OFFSET_FULL

```
#define GTK_LEVEL_BAR_OFFSET_FULL "full"
```

The name used for the stock full offset included by GtkLevelBar.

Since: 3.20

---

enum GtkLevelBarMode

Describes how GtkLevelBar contents should be rendered. Note that this enumeration could be extended with additional modes in the future.

Members

GTK_LEVEL_BAR_MODE_CONTINUOUS	the bar has a continuous mode
GTK_LEVEL_BAR_MODE_DISCRETE	the bar has a discrete mode

Since: 3.6

---

struct GtkLevelBar

```
struct GtkLevelBar;
```

Property Details

The “inverted” property

```
  “inverted”                 gboolean
```

Level bars normally grow from top to bottom or left to right. Inverted level bars grow in the opposite direction.

Flags: Read / Write

Default value: FALSE

Since: 3.8

The “max-value” property

```
  “max-value”                gdouble
```

The “max-value” property determaxes the maximum value of the interval that can be displayed by the bar.

Flags: Read / Write

Allowed values: >= 0

Default value: 1

Since: 3.6

The “min-value” property

```
  “min-value”                gdouble
```

The “min-value” property determines the minimum value of the interval that can be displayed by the bar.

Flags: Read / Write

Allowed values: >= 0

Default value: 0

Since: 3.6

The “mode” property

```
  “mode”                     GtkLevelBarMode
```

The “mode” property determines the way GtkLevelBar interprets the value properties to draw the level fill area. Specifically, when the value is GTK_LEVEL_BAR_MODE_CONTINUOUS, GtkLevelBar will draw a single block representing the current value in that area; when the value is GTK_LEVEL_BAR_MODE_DISCRETE,

the widget will draw a succession of separate blocks filling the draw area, with the number of blocks being equal to the units separating the integral roundings of “min-value” and “max-value”.

Flags: Read / Write

Default value: GTK_LEVEL_BAR_MODE_CONTINUOUS

Since: 3.6

The “value” property

“value” gdouble

The “value” property determines the currently filled value of the level bar.

Flags: Read / Write

Allowed values: >= 0

Default value: 0

Since: 3.6

Style Property Details

The “min-block-height” style property

“min-block-height” gint

The min-block-height style property determines the minimum height for blocks filling the GtkLevelBar widget.

`GtkLevelBar:min-block-height` has been deprecated since version 3.20 and should not be used in newly-written code.

Use the standard min-width/min-height CSS properties on the block elements; the value of this style property is ignored.

Flags: Read / Write

Allowed values: >= 1

Default value: 3

Since: 3.6

The “min-block-width” style property

“min-block-width” gint

The min-block-width style property determines the minimum width for blocks filling the GtkLevelBar widget.

`GtkLevelBar:min-block-width` has been deprecated since version 3.20 and should not be used in newly-written code.

Use the standard min-width/min-height CSS properties on the block elements; the value of this style property is ignored.

Flags: Read / Write

Allowed values: >= 1

Default value: 3

Since: 3.6

Signal Details

The `"offset-changed"` signal

```
void
user_function (GtkLevelBar *self,
               gchar       *name,
               gpointer     user_data)
```

Emitted when an offset specified on the bar changes value as an effect to `gtk_level_bar_add_offset_value()` being called.

The signal supports detailed connections; you can connect to the detailed signal "changed::x" in order to only receive callbacks when the value of offset "x" changes.

Parameters

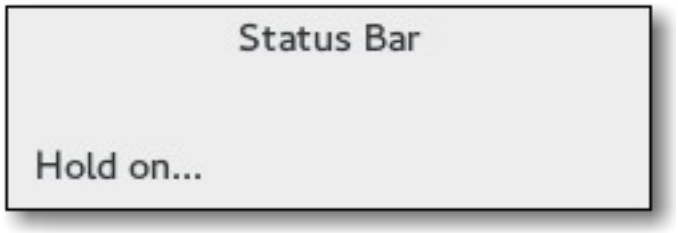
self	a GtkLevelBar
name	the name of the offset that changed value
user_data	user data set when the signal handler was connected.

Flags: Has Details

Since: 3.6

GtkStatusbar

GtkStatusbar — Report messages of minor importance to the user



Functions

GtkWidget *	gtk_statusbar_new ()
guint	gtk_statusbar_get_context_id ()
guint	gtk_statusbar_push ()
void	gtk_statusbar_pop ()
void	gtk_statusbar_remove ()
void	gtk_statusbar_remove_all ()
GtkWidget *	gtk_statusbar_get_message_area ()

Style Properties

GtkShadowType	shadow-type	Read

Signals

void	text-popped	Run Last
void	text-pushed	Run Last

Types and Values

struct	GtkStatusbar

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkContainer
            └── GtkBox
                └── GtkStatusbar
```

Implemented Interfaces

GtkStatusbar implements AtkImplementorIface, GtkBuildable and GtkOrientable.

Includes

```
#include <gtk/gtk.h>
```

Description

A GtkStatusbar is usually placed along the bottom of an application's main GtkWindow. It may provide a regular commentary of the application's status (as is usually the case in a web browser, for example), or may be used to simply output a message when the status changes, (when an upload is complete in an FTP client, for example).

Status bars in GTK+ maintain a stack of messages. The message at the top of the each bar’s stack is the one that will currently be displayed.

Any messages added to a statusbar's stack must specify a context id that is used to uniquely identify the source of a message. This context id can be generated by `gtk_statusbar_get_context_id()`, given a message and the statusbar that it will be added to. Note that messages are stored in a stack, and when choosing which message to display, the stack structure is adhered to, regardless of the context identifier of a message.

One could say that a statusbar maintains one stack of messages for display purposes, but allows multiple message producers to maintain sub-stacks of the messages they produced (via context ids).

Status bars are created using `gtk_statusbar_new()`.

Messages are added to the bar's stack with `gtk_statusbar_push()`.

The message at the top of the stack can be removed using `gtk_statusbar_pop()`. A message can be removed from anywhere in the stack if its message id was recorded at the time it was added. This is done using `gtk_statusbar_remove()`.

CSS node

GtkStatusbar has a single CSS node with name statusbar.

Functions

gtk_statusbar_new ()

```
GtkWidget *
gtk_statusbar_new (void);
```

Creates a new GtkStatusbar ready for messages.

Returns

the new GtkStatusbar

---

gtk_statusbar_get_context_id ()

```
guint
gtk_statusbar_get_context_id (GtkStatusbar *statusbar,
                              const gchar *context_description);
```

Returns a new context identifier, given a description of the actual context. Note that the description is not shown in the UI.

Parameters

statusbar	a GtkStatusbar
context_description	textual description of what context the new message is being used in

Returns

an integer id

---

gtk_statusbar_push ()

```
guint
gtk_statusbar_push (GtkStatusbar *statusbar,
                    guint context_id,
                    const gchar *text);
```

Pushes a new message onto a statusbar's stack.

Parameters

statusbar	a GtkStatusbar
context_id	the message's context id, as returned by gtk_statusbar_get_context_id()
text	the message to add to the statusbar

Returns

a message id that can be used with `gtk_statusbar_remove()`.

---

gtk_statusbar_pop ()

```
void
gtk_statusbar_pop (GtkStatusbar *statusbar,
                   guint context_id);
```

Removes the first message in the GtkStatusbar's stack with the given context id.

Note that this may not change the displayed message, if the message at the top of the stack has a different context id.

Parameters

statusbar	a GtkStatusbar
context_id	a context identifier

---

gtk_statusbar_remove ()

```
void
gtk_statusbar_remove (GtkStatusbar *statusbar,
                      guint context_id,
                      guint message_id);
```

Forces the removal of a message from a statusbar's stack. The exact `context_id` and `message_id` must be

specified.

Parameters

statusbar	a GtkStatusbar
context_id	a context identifier
message_id	a message identifier, as returned by `gtk_statusbar_push()`

---

gtk_statusbar_remove_all ()

```
void
gtk_statusbar_remove_all (GtkStatusbar *statusbar,
                          guint context_id);
```

Forces the removal of all messages from a statusbar's stack with the exact `context_id` .

Parameters

statusbar	a GtkStatusbar
context_id	a context identifier

Since: 2.22

---

gtk_statusbar_get_message_area ()

```
GtkWidget *
gtk_statusbar_get_message_area (GtkStatusbar *statusbar);
```

Retrieves the box containing the label widget.

Parameters

statusbar	a GtkStatusbar

Returns

a GtkBox.

[type Gtk.Box][transfer none]

Since: 2.20

Types and Values

struct GtkStatusbar

```
struct GtkStatusbar;
```

Style Property Details

The “shadow-type” style property

```
  “shadow-type”              GtkShadowType
```

The style of the bevel around the statusbar text.

`GtkStatusbar:shadow-type` has been deprecated since version 3.20 and should not be used in newly-written code.

Use CSS properties to determine the appearance, the value of this style property is ignored.

Flags: Read

Default value: GTK_SHADOW_IN

Signal Details

The “text-popped” signal

```
void
user_function (GtkStatusbar *statusbar,
               guint         context_id,
               gchar        *text,
               gpointer      user_data)
```

Is emitted whenever a new message is popped off a statusbar's stack.

Parameters

statusbar	the object which received the signal
context_id	the context id of the relevant message/statusbar
text	the message that was just popped
user_data	user data set when the signal handler was connected.

Flags: Run Last

The “text-pushed” signal

```
void
user_function (GtkStatusbar *statusbar,
               guint         context_id,
               gchar        *text,
               gpointer      user_data)
```

Is emitted whenever a new message gets pushed onto a statusbar's stack.

Parameters

statusbar	the object which received the signal
context_id	the context id of the relevant message/statusbar
text	the message that was pushed
user_data	user data set when the signal handler was connected.

Flags: Run Last

GtkAccelLabel

GtkAccelLabel — A label which displays an accelerator key on the right of the text

Functions

GtkWidget *	gtk_accel_label_new ()
void	gtk_accel_label_set_accel_closure ()
GtkWidget *	gtk_accel_label_get_accel_widget ()
void	gtk_accel_label_set_accel_widget ()
guint	gtk_accel_label_get_accel_width ()
void	gtk_accel_label_set_accel ()
void	gtk_accel_label_get_accel ()
gboolean	gtk_accel_label_refetch ()

Properties

GClosure *	accel-closure	Read / Write
GtkWidget *	accel-widget	Read / Write

Types and Values

struct	GtkAccelLabel

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
```

```
        └── GtkMisc
            └── GtkLabel
                └── GtkAccelLabel
```

Implemented Interfaces

GtkAccelLabel implements AtkImplementorIface and GtkBuildable.

Includes

```
#include <gtk/gtk.h>
```

Description

The GtkAccelLabel widget is a subclass of GtkLabel that also displays an accelerator key on the right of the label text, e.g. “Ctrl+S”. It is commonly used in menus to show the keyboard short-cuts for commands.

The accelerator key to display is typically not set explicitly (although it can be, with `gtk_accel_label_set_accel()`). Instead, the GtkAccelLabel displays the accelerators which have been added to a particular widget. This widget is set by calling `gtk_accel_label_set_accel_widget()`.

For example, a GtkMenuItem widget may have an accelerator added to emit the “activate” signal when the “Ctrl+S” key combination is pressed. A GtkAccelLabel is created and added to the GtkMenuItem, and `gtk_accel_label_set_accel_widget()` is called with the GtkMenuItem as the second argument. The GtkAccelLabel will now display “Ctrl+S” after its label.

Note that creating a GtkMenuItem with `gtk_menu_item_new_with_label()` (or one of the similar functions for GtkCheckMenuItem and GtkRadioMenuItem) automatically adds a GtkAccelLabel to the GtkMenuItem and calls `gtk_accel_label_set_accel_widget()` to set it up for you.

A GtkAccelLabel will only display accelerators which have `GTK_ACCEL_VISIBLE` set (see GtkAccelFlags). A GtkAccelLabel can display multiple accelerators and even signal names, though it is almost always used to display just one accelerator key.

Creating a simple menu item with an accelerator key.

```
1   GtkWidget *window = gtk_window_new (GTK_WINDOW_TOPLEVEL);
2   GtkWidget *menu = gtk_menu_new ();
3   GtkWidget *save_item;
4   GtkAccelGroup *accel_group;
5
6   // Create a GtkAccelGroup and add it to the window.
7   accel_group = gtk_accel_group_new ();
8   gtk_window_add_accel_group (GTK_WINDOW (window), accel_group);
9
10  // Create the menu item using the convenience function.
11  save_item = gtk_menu_item_new_with_label ("Save");
12  gtk_widget_show (save_item);
13  gtk_container_add (GTK_CONTAINER (menu), save_item);
14
15  // Now add the accelerator to the GtkMenuItem. Note that since we
16  // called gtk_menu_item_new_with_label() to create the GtkMenuItem
17  // the GtkAccelLabel is automatically set up to display the
18  // GtkMenuItem accelerators. We just need to make sure we use
```

```
19   // GTK_ACCEL_VISIBLE here.
20   gtk_widget_add_accelerator (save_item, "activate", accel_group,
21                               GDK_KEY_s, GDK_CONTROL_MASK, GTK_ACCEL_VISIBLE);
```

CSS nodes

```
1                                                  label
2                                                  └── accelerator
```

Like GtkLabel, GtkAccelLabel has a main CSS node with the name label. It adds a subnode with name accelerator.

Functions

gtk_accel_label_new ()

```
GtkWidget *
gtk_accel_label_new (const gchar *string);
```

Creates a new GtkAccelLabel.

Parameters

string	the label string. Must be non-`NULL`.	

Returns

a new GtkAccelLabel.

---

gtk_accel_label_set_accel_closure ()

```
void
gtk_accel_label_set_accel_closure (GtkAccelLabel *accel_label,
                                   GClosure *accel_closure);
```

Sets the closure to be monitored by this accelerator label. The closure must be connected to an accelerator group; see `gtk_accel_group_connect()`. Passing `NULL` for `accel_closure` will dissociate `accel_label` from its current closure, if any.

Parameters

accel_label	a GtkAccelLabel	
accel_closure	the closure to monitor for accelerator changes, or `NULL`.	[nullable]

---

gtk_accel_label_get_accel_widget ()

```
GtkWidget *
gtk_accel_label_get_accel_widget (GtkAccelLabel *accel_label);
```

Fetches the widget monitored by this accelerator label. See `gtk_accel_label_set_accel_widget()`.

Parameters

accel_label	a GtkAccelLabel

Returns

the object monitored by the accelerator label, or `NULL`.

[nullable][transfer none]

---

gtk_accel_label_set_accel_widget ()

```
void
gtk_accel_label_set_accel_widget (GtkAccelLabel *accel_label,
                                  GtkWidget *accel_widget);
```

Sets the widget to be monitored by this accelerator label. Passing `NULL` for `accel_widget` will dissociate `accel_label` from its current widget, if any.

Parameters

accel_label	a GtkAccelLabel
accel_widget	the widget to be monitored, or `NULL`. [nullable]

---

gtk_accel_label_get_accel_width ()

```
guint
gtk_accel_label_get_accel_width (GtkAccelLabel *accel_label);
```

Returns the width needed to display the accelerator key(s). This is used by menus to align all of the GtkMenuItem widgets, and shouldn't be needed by applications.

Parameters

accel_label	a GtkAccelLabel.

Returns

the width needed to display the accelerator key(s).

---

gtk_accel_label_set_accel ()

```
void
gtk_accel_label_set_accel (GtkAccelLabel *accel_label,
                           guint accelerator_key,
                           GdkModifierType accelerator_mods);
```

Manually sets a keyval and modifier mask as the accelerator rendered by `accel_label`.

If a keyval and modifier are explicitly set then these values are used regardless of any associated accel closure or widget.

Providing an `accelerator_key` of 0 removes the manual setting.

Parameters

accel_label	a GtkAccelLabel	
accelerator_key	a keyval, or 0	
accelerator_mods	the modifier mask for the accel	

Since: 3.6

---

gtk_accel_label_get_accel ()

```
void
gtk_accel_label_get_accel (GtkAccelLabel *accel_label,
                           guint *accelerator_key,
                           GdkModifierType *accelerator_mods);
```

Gets the keyval and modifier mask set with `gtk_accel_label_set_accel()`.

Parameters

accel_label	a GtkAccelLabel	
accelerator_key	return location for the keyval.	[out]
accelerator_mods	return location for the modifier mask.	[out]

Since: 3.12

---

gtk_accel_label_refetch ()

```
gboolean
gtk_accel_label_refetch (GtkAccelLabel *accel_label);
```

Recreates the string representing the accelerator keys. This should not be needed since the string is automatically updated whenever accelerators are added or removed from the associated widget.

Parameters

accel_label	a GtkAccelLabel.

Returns

always returns FALSE.

Types and Values

struct GtkAccelLabel

```
struct GtkAccelLabel;
```

The GtkAccelLabel contains private data only, and should be accessed using the functions below.

Property Details

The “accel-closure” property

```
  “accel-closure”            GClosure *
```

The closure to be monitored for accelerator changes.

Flags: Read / Write

The “accel-widget” property

```
  “accel-widget”             GtkWidget *
```

The widget to be monitored for accelerator changes.

Flags: Read / Write

See Also

GtkAccelGroup

Buttons and Toggles

GtkButton — A widget that emits a signal when clicked on

GtkCheckButton — Create widgets with a discrete toggle button

GtkRadioButton — A choice from multiple check buttons

GtkToggleButton — Create buttons which retain their state

GtkLinkButton — Create buttons bound to a URL

GtkMenuButton — A widget that shows a popup when clicked on

GtkSwitch — A “light switch” style toggle

GtkScaleButton — A button which pops up a scale

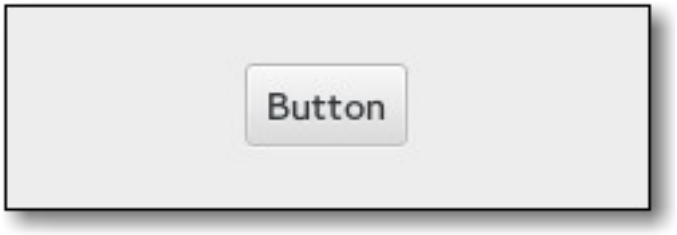
GtkVolumeButton — A button which pops up a volume control

GtkLockButton — A widget to unlock or lock privileged operations

GtkModelButton — A button that uses a GAction as model

GtkButton

GtkButton — A widget that emits a signal when clicked on



Functions

GtkWidget *	gtk_button_new ()
GtkWidget *	gtk_button_new_with_label ()
GtkWidget *	gtk_button_new_with_mnemonic ()
GtkWidget *	gtk_button_new_from_icon_name ()
GtkWidget *	gtk_button_new_from_stock ()
void	gtk_button_pressed ()
void	gtk_button_released ()
void	gtk_button_clicked ()
void	gtk_button_enter ()
void	gtk_button_leave ()
void	gtk_button_set_relief ()
GtkReliefStyle	gtk_button_get_relief ()
const gchar *	gtk_button_get_label ()
void	gtk_button_set_label ()
gboolean	gtk_button_get_use_stock ()
void	gtk_button_set_use_stock ()
gboolean	gtk_button_get_use_underline ()
void	gtk_button_set_use_underline ()
void	gtk_button_set_focus_on_click ()
gboolean	gtk_button_get_focus_on_click ()
void	gtk_button_set_alignment ()
void	gtk_button_get_alignment ()
void	gtk_button_set_image ()
GtkWidget *	gtk_button_get_image ()
void	gtk_button_set_image_position ()
GtkPositionType	gtk_button_get_image_position ()
void	gtk_button_set_always_show_image ()
gboolean	gtk_button_get_always_show_image ()
GdkWindow *	gtk_button_get_event_window ()

Properties

gboolean	always-show-image	Read / Write / Construct
GtkWidget *	image	Read / Write
GtkPositionType	image-position	Read / Write
gchar *	label	Read / Write / Construct
GtkReliefStyle	relief	Read / Write
gboolean	use-stock	Read / Write / Construct
gboolean	use-underline	Read / Write / Construct
gfloat	xalign	Read / Write
gfloat	yalign	Read / Write

Style Properties

gint	child-displacement-x	Read
gint	child-displacement-y	Read
GtkBorder *	default-border	Read
GtkBorder *	default-outside-border	Read
gboolean	displace-focus	Read
gint	image-spacing	Read
GtkBorder *	inner-border	Read

Signals

void	activate	Action
void	clicked	Action
void	enter	Run First
void	leave	Run First
void	pressed	Run First
void	released	Run First

Types and Values

struct	GtkButton
struct	GtkButtonClass

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkContainer
            └── GtkBin
                └── GtkButton
                    ├── GtkToggleButton
                    ├── GtkColorButton
                    ├── GtkFontButton
                    ├── GtkLinkButton
                    ├── GtkLockButton
                    ├── GtkModelButton
                    └── GtkScaleButton
```

Implemented Interfaces

GtkButton implements AtkImplementorIface, GtkBuildable, GtkActionable and GtkActivatable.

Includes

```
#include <gtk/gtk.h>
```

Description

The GtkButton widget is generally used to trigger a callback function that is called when the button is pressed. The various signals and how to use them are outlined below.

The GtkButton widget can hold any valid child widget. That is, it can hold almost any other standard GtkWidget. The most commonly used child is the GtkLabel.

CSS nodes

GtkButton has a single CSS node with name button. The node will get the style classes .image-button or .text-button, if the content is just an image or label, respectively. It may also receive the .flat style class.

Other style classes that are commonly used with GtkButton include .suggested-action and .destructive-action. In special cases, buttons can be made round by adding the .circular style class.

Button-like widgets like GtkToggleButton, GtkMenuButton, GtkVolumeButton, GtkLockButton, GtkColorButton, GtkFontButton or GtkFileChooserButton use style classes such as .toggle, .popup, .scale, .lock, .color, .font, .file to differentiate themselves from a plain GtkButton.

Functions

gtk_button_new ()

```
GtkWidget *
gtk_button_new (void);
```

Creates a new GtkButton widget. To add a child widget to the button, use `gtk_container_add()`.

Returns

The newly created GtkButton widget.

---

gtk_button_new_with_label ()

```
GtkWidget *
gtk_button_new_with_label (const gchar *label);
```

Creates a GtkButton widget with a GtkLabel child containing the given text.

Parameters

label	The text you want the GtkLabel to hold.

Returns

The newly created GtkButton widget.

---

gtk_button_new_with_mnemonic ()

```
GtkWidget *
gtk_button_new_with_mnemonic (const gchar *label);
```

Creates a new GtkButton containing a label. If characters in `label` are preceded by an underscore, they are underlined. If you need a literal underscore character in a label, use "__" (two underscores). The first underlined character represents a keyboard accelerator called a mnemonic. Pressing Alt and that key activates the button.

Parameters

label	The text of the button, with an underscore in front of the mnemonic character

Returns

a new GtkButton

---

gtk_button_new_from_icon_name ()

```
GtkWidget *
gtk_button_new_from_icon_name (const gchar *icon_name,
                               GtkIconSize size);
```

Creates a new button containing an icon from the current icon theme.

If the icon name isn't known, a "broken image" icon will be displayed instead. If the current icon theme is changed, the icon will be updated appropriately.

This function is a convenience wrapper around `gtk_button_new()` and `gtk_button_set_image()`.

Parameters

icon_name	an icon name or `NULL`.	[nullable]
size	an icon size (GtkIconSize).	[type int]

Returns

a new GtkButton displaying the themed icon

Since: 3.10

---

gtk_button_new_from_stock ()

```
GtkWidget *
gtk_button_new_from_stock (const gchar *stock_id);
```

`gtk_button_new_from_stock` has been deprecated since version 3.10 and should not be used in newly-written code.

Stock items are deprecated. Use `gtk_button_new_with_label()` instead.

Creates a new GtkButton containing the image and text from a stock item. Some stock ids have preprocessor macros like GTK_STOCK_OK and GTK_STOCK_APPLY.

If `stock_id` is unknown, then it will be treated as a mnemonic label (as for `gtk_button_new_with_mnemonic()`).

Parameters

stock_id	the name of the stock item

Returns

a new GtkButton

---

gtk_button_pressed ()

```
void
gtk_button_pressed (GtkButton *button);
```

`gtk_button_pressed` has been deprecated since version 2.20 and should not be used in newly-written code.

Use the "button-press-event" signal.

Emits a "pressed" signal to the given GtkButton.

Parameters

button	The GtkButton you want to send the signal to.

---

gtk_button_released ()

```
void
```

```
gtk_button_released (GtkButton *button);
```

`gtk_button_released` has been deprecated since version 2.20 and should not be used in newly-written code.

Use the "button-release-event" signal.

Emits a "released" signal to the given GtkButton.

Parameters

button	The GtkButton you want to send the signal to.

---

gtk_button_clicked ()

```
void
gtk_button_clicked (GtkButton *button);
```

Emits a "clicked" signal to the given GtkButton.

Parameters

button	The GtkButton you want to send the signal to.

---

gtk_button_enter ()

```
void
gtk_button_enter (GtkButton *button);
```

`gtk_button_enter` has been deprecated since version 2.20 and should not be used in newly-written code.

Use the "enter-notify-event" signal.

Emits a "enter" signal to the given GtkButton.

Parameters

button	The GtkButton you want to send the signal to.

---

gtk_button_leave ()

```
void
gtk_button_leave (GtkButton *button);
```

`gtk_button_leave` has been deprecated since version 2.20 and should not be used in newly-written code.

Use the "leave-notify-event" signal.

Emits a “leave” signal to the given GtkButton.

Parameters

button	The GtkButton you want to send the signal to.

---

gtk_button_set_relief ()

```
void
gtk_button_set_relief (GtkButton *button,
                       GtkReliefStyle relief);
```

Sets the relief style of the edges of the given GtkButton widget. Two styles exist, GTK_RELIEF_NORMAL and GTK_RELIEF_NONE. The default style is, as one can guess, GTK_RELIEF_NORMAL. The deprecated value GTK_RELIEF_HALF behaves the same as GTK_RELIEF_NORMAL.

Parameters

button	The GtkButton you want to set relief styles of
relief	The GtkReliefStyle as described above

---

gtk_button_get_relief ()

```
GtkReliefStyle
gtk_button_get_relief (GtkButton *button);
```

Returns the current relief style of the given GtkButton.

Parameters

button	The GtkButton you want the GtkReliefStyle from.

Returns

The current GtkReliefStyle

---

gtk_button_get_label ()

```
const gchar *
gtk_button_get_label (GtkButton *button);
```

Fetches the text from the label of the button, as set by gtk_button_set_label(). If the label text has not been

set the return value will be NULL. This will be the case if you create an empty button with gtk_button_new() to use as a container.

Parameters

button	a GtkButton

Returns

The text of the label widget. This string is owned by the widget and must not be modified or freed.

---

gtk_button_set_label ()

```
void
gtk_button_set_label (GtkButton *button,
                      const gchar *label);
```

Sets the text of the label of the button to str . This text is also used to select the stock item if gtk_button_set_use_stock() is used.

This will also clear any previously set labels.

Parameters

button	a GtkButton
label	a string

---

gtk_button_get_use_stock ()

```
gboolean
gtk_button_get_use_stock (GtkButton *button);
```

gtk_button_get_use_stock has been deprecated since version 3.10 and should not be used in newly-written code.

Returns whether the button label is a stock item.

Parameters

button	a GtkButton

Returns

TRUE if the button label is used to select a stock item instead of being used directly as the label text.

---

gtk_button_set_use_stock ()

```
void
gtk_button_set_use_stock (GtkButton *button,
                          gboolean use_stock);
```

`gtk_button_set_use_stock` has been deprecated since version 3.10 and should not be used in newly-written code.

If `TRUE`, the label set on the button is used as a stock id to select the stock item for the button.

Parameters

button	a GtkButton
use_stock	`TRUE` if the button should use a stock item

---

gtk_button_get_use_underline ()

```
gboolean
gtk_button_get_use_underline (GtkButton *button);
```

Returns whether an embedded underline in the button label indicates a mnemonic. See `gtk_button_set_use_underline()`.

Parameters

button	a GtkButton

Returns

`TRUE` if an embedded underline in the button label indicates the mnemonic accelerator keys.

---

gtk_button_set_use_underline ()

```
void
gtk_button_set_use_underline (GtkButton *button,
                              gboolean use_underline);
```

If true, an underline in the text of the button label indicates the next character should be used for the mnemonic accelerator key.

Parameters

button	a GtkButton
use_underline	`TRUE` if underlines in the text indicate mnemonics

---

gtk_button_set_focus_on_click ()

```
void
gtk_button_set_focus_on_click (GtkButton *button,
                               gboolean focus_on_click);
```

`gtk_button_set_focus_on_click` has been deprecated since version 3.20 and should not be used in newly-written code.

Use `gtk_widget_set_focus_on_click()` instead

Sets whether the button will grab focus when it is clicked with the mouse. Making mouse clicks not grab focus is useful in places like toolbars where you don’t want the keyboard focus removed from the main area of the application.

Parameters

button	a GtkButton
focus_on_click	whether the button grabs focus when clicked with the mouse

Since: 2.4

---

gtk_button_get_focus_on_click ()

```
gboolean
gtk_button_get_focus_on_click (GtkButton *button);
```

`gtk_button_get_focus_on_click` has been deprecated since version 3.20 and should not be used in newly-written code.

Use `gtk_widget_get_focus_on_click()` instead

Returns whether the button grabs focus when it is clicked with the mouse. See `gtk_button_set_focus_on_click()`.

Parameters

button	a GtkButton

Returns

`TRUE` if the button grabs focus when it is clicked with the mouse.

Since: 2.4

---

gtk_button_set_alignment ()

```
void
gtk_button_set_alignment (GtkButton *button,
                          gfloat xalign,
                          gfloat yalign);
```

`gtk_button_set_alignment` has been deprecated since version 3.14 and should not be used in newly-written code.

Access the child widget directly if you need to control its alignment.

Sets the alignment of the child. This property has no effect unless the child is a GtkMisc or a GtkAlignment.

Parameters

button	a GtkButton	
xalign	the horizontal position of the child, 0.0 is left aligned, 1.0 is right aligned	
yalign	the vertical position of the child, 0.0 is top aligned, 1.0 is bottom aligned	

Since: 2.4

---

gtk_button_get_alignment ()

```
void
gtk_button_get_alignment (GtkButton *button,
                          gfloat *xalign,
                          gfloat *yalign);
```

`gtk_button_get_alignment` has been deprecated since version 3.14 and should not be used in newly-written code.

Access the child widget directly if you need to control its alignment.

Gets the alignment of the child in the button.

Parameters

button	a GtkButton	
xalign	return location for horizontal alignment.	[out]
yalign	return location for vertical alignment.	[out]

Since: 2.4

---

gtk_button_set_image ()

```
void
gtk_button_set_image (GtkButton *button,
                      GtkWidget *image);
```

Set the image of `button` to the given widget. The image will be displayed if the label text is `NULL` or if "always-show-image" is `TRUE`. You don't have to call `gtk_widget_show()` on `image` yourself.

Parameters

button	a GtkButton	
image	a widget to set as the image for the button, or `NULL` to unset.	[nullable]

Since: 2.6

---

gtk_button_get_image ()

```
GtkWidget *
gtk_button_get_image (GtkButton *button);
```

Gets the widget that is currenty set as the image of `button` . This may have been explicitly set by `gtk_button_set_image()` or constructed by `gtk_button_new_from_stock()`.

Parameters

button	a GtkButton

Returns

a GtkWidget or `NULL` in case there is no image.

[nullable][transfer none]

Since: 2.6

---

gtk_button_set_image_position ()

```
void
gtk_button_set_image_position (GtkButton *button,
                               GtkPositionType position);
```

Sets the position of the image relative to the text inside the button.

Parameters

button	a GtkButton	
position	the position	

Since: 2.10

---

gtk_button_get_image_position ()

```
GtkPositionType
gtk_button_get_image_position (GtkButton *button);
```

Gets the position of the image relative to the text inside the button.

Parameters

button	a GtkButton

Returns

the position

Since: 2.10

---

gtk_button_set_always_show_image ()

```
void
gtk_button_set_always_show_image (GtkButton *button,
                                  gboolean always_show);
```

If TRUE, the button will ignore the "gtk-button-images" setting and always show the image, if available.

Use this property if the button would be useless or hard to use without the image.

Parameters

button	a GtkButton
always_show	TRUE if the menuitem should always show the image

Since: 3.6

---

gtk_button_get_always_show_image ()

```
gboolean
gtk_button_get_always_show_image (GtkButton *button);
```

Returns whether the button will ignore the "gtk-button-images" setting and always show the image, if available.

Parameters

button	a GtkButton

Returns

TRUE if the button will always show the image

Since: 3.6

---

gtk_button_get_event_window ()

```
GdkWindow *
gtk_button_get_event_window (GtkButton *button);
```

Returns the button’s event window if it is realized, `NULL` otherwise. This function should be rarely needed.

Parameters

button	a GtkButton

Returns

`button` ’s event window.

[transfer none]

Since: 2.22

Types and Values

struct GtkButton

```
struct GtkButton;
```

---

struct GtkButtonClass

```
struct GtkButtonClass {
  GtkBinClass        parent_class;


  void (* pressed)  (GtkButton *button);
  void (* released) (GtkButton *button);
  void (* clicked)  (GtkButton *button);
  void (* enter)    (GtkButton *button);
  void (* leave)    (GtkButton *button);
  void (* activate) (GtkButton *button);
};
```

Members

`pressed` ()	Signal emitted when the button is pressed. Deprecated: 2.8.
`released` ()	Signal emitted when the button is released. Deprecated: 2.8.
`clicked` ()	Signal emitted when the button has been activated (pressed and released).
`enter` ()	Signal emitted when the pointer enters the button. Deprecated: 2.8.
`leave` ()	Signal emitted when the pointer leaves the button. Deprecated: 2.8.
`activate` ()	Signal that causes the button to

animate press then release. Applications should never connect to this signal, but use the `clicked` signal.

Property Details

The `“always-show-image”` property

`“always-show-image”` `gboolean`

If `TRUE`, the button will ignore the “gtk-button-images” setting and always show the image, if available.

Use this property if the button would be useless or hard to use without the image.

Flags: Read / Write / Construct

Default value: FALSE

Since: 3.6

The `“image”` property

`“image”` `GtkWidget *`

The child widget to appear next to the button text.

Flags: Read / Write

Since: 2.6

The `“image-position”` property

`“image-position”` `GtkPositionType`

The position of the image relative to the text inside the button.

Flags: Read / Write

Default value: GTK_POS_LEFT

Since: 2.10

The `“label”` property

`“label”` `gchar *`

Text of the label widget inside the button, if the button contains a label widget.

Flags: Read / Write / Construct

Default value: NULL

The “relief” property

“relief” GtkReliefStyle

The border relief style.

Flags: Read / Write

Default value: GTK_RELIEF_NORMAL

The “use-stock” property

“use-stock” gboolean

If set, the label is used to pick a stock item instead of being displayed.

GtkButton:use-stock has been deprecated since version 3.10 and should not be used in newly-written code.

Flags: Read / Write / Construct

Default value: FALSE

The “use-underline” property

“use-underline” gboolean

If set, an underline in the text indicates the next character should be used for the mnemonic accelerator key.

Flags: Read / Write / Construct

Default value: FALSE

The “xalign” property

“xalign” gfloat

If the child of the button is a GtkMisc or GtkAlignment, this property can be used to control its horizontal alignment. 0.0 is left aligned, 1.0 is right aligned.

GtkButton:xalign has been deprecated since version 3.14 and should not be used in newly-written code.

Access the child widget directly if you need to control its alignment.

Flags: Read / Write

Allowed values: [0,1]

Default value: 0.5

Since: 2.4

The “yalign” property

“yalign” gfloat

If the child of the button is a GtkMisc or GtkAlignment, this property can be used to control its vertical alignment. 0.0 is top aligned, 1.0 is bottom aligned.

GtkButton:yalign has been deprecated since version 3.14 and should not be used in newly-written code.

Access the child widget directly if you need to control its alignment.

Flags: Read / Write

Allowed values: [0,1]

Default value: 0.5

Since: 2.4

Style Property Details

The “child-displacement-x” style property

“child-displacement-x” gint

How far in the x direction to move the child when the button is depressed.

GtkButton:child-displacement-x has been deprecated since version 3.20 and should not be used in newly-written code.

Use CSS margins and padding instead; the value of this style property is ignored.

Flags: Read

Default value: 0

The “child-displacement-y” style property

“child-displacement-y” gint

How far in the y direction to move the child when the button is depressed.

GtkButton:child-displacement-y has been deprecated since version 3.20 and should not be used in newly-written code.

Use CSS margins and padding instead; the value of this style property is ignored.

Flags: Read

Default value: 0

The “default-border” style property

“default-border” GtkBorder *

The "default-border" style property defines the extra space to add around a button that can become the default widget of its window. For more information about default widgets, see `gtk_widget_grab_default()`.

`GtkButton:default-border` has been deprecated since version 3.14 and should not be used in newly-written code.

Use CSS margins and padding instead; the value of this style property is ignored.

Flags: Read

The `"default-outside-border"` style property

`"default-outside-border"` `GtkBorder *`

The "default-outside-border" style property defines the extra outside space to add around a button that can become the default widget of its window. Extra outside space is always drawn outside the button border. For more information about default widgets, see `gtk_widget_grab_default()`.

`GtkButton:default-outside-border` has been deprecated since version 3.14 and should not be used in newly-written code.

Use CSS margins and padding instead; the value of this style property is ignored.

Flags: Read

The `"displace-focus"` style property

`"displace-focus"` `gboolean`

Whether the child_displacement_x/child_displacement_y properties should also affect the focus rectangle.

`GtkButton:displace-focus` has been deprecated since version 3.20 and should not be used in newly-written code.

Use CSS margins and padding instead; the value of this style property is ignored.

Flags: Read

Default value: FALSE

Since: 2.6

The `"image-spacing"` style property

`"image-spacing"` `gint`

Spacing in pixels between the image and label.

Flags: Read

Allowed values: >= 0

Default value: 2

The “inner-border” style property

```
  “inner-border”             GtkBorder *
```

Sets the border between the button edges and child.

`GtkButton:inner-border` has been deprecated since version 3.4 and should not be used in newly-written code.

Use the standard border and padding CSS properties; the value of this style property is ignored.

Flags: Read

Since: 2.10

Signal Details

The “activate” signal

```
void
user_function (GtkButton *widget,
               gpointer   user_data)
```

The ::activate signal on GtkButton is an action signal and emitting it causes the button to animate press then release. Applications should never connect to this signal, but use the “clicked” signal.

Parameters

widget	the object which received the signal.
user_data	user data set when the signal handler was connected.

Flags: Action

The “clicked” signal

```
void
user_function (GtkButton *button,
               gpointer   user_data)
```

Emitted when the button has been activated (pressed and released).

Parameters

button	the object that received the signal
user_data	user data set when the signal handler was connected.

Flags: Action

The “enter” signal

```
void
user_function (GtkButton *button,
               gpointer   user_data)
```

Emitted when the pointer enters the button.

`GtkButton::enter` has been deprecated since version 2.8 and should not be used in newly-written code.

Use the “enter-notify-event” signal.

Parameters

button	the object that received the signal
user_data	user data set when the signal handler was connected.

Flags: Run First

The “leave” signal

```
void
user_function (GtkButton *button,
               gpointer   user_data)
```

Emitted when the pointer leaves the button.

`GtkButton::leave` has been deprecated since version 2.8 and should not be used in newly-written code.

Use the “leave-notify-event” signal.

Parameters

button	the object that received the signal
user_data	user data set when the signal handler was connected.

Flags: Run First

The “pressed” signal

```
void
user_function (GtkButton *button,
               gpointer   user_data)
```

Emitted when the button is pressed.

`GtkButton::pressed` has been deprecated since version 2.8 and should not be used in newly-written code.

Use the “button-press-event” signal.

Parameters

button	the object that received the signal
user_data	user data set when the signal handler was connected.

Flags: Run First

The `"released"` signal

```
void
user_function (GtkButton *button,
               gpointer   user_data)
```

Emitted when the button is released.

`GtkButton::released` has been deprecated since version 2.8 and should not be used in newly-written code.

Use the "button-release-event" signal.

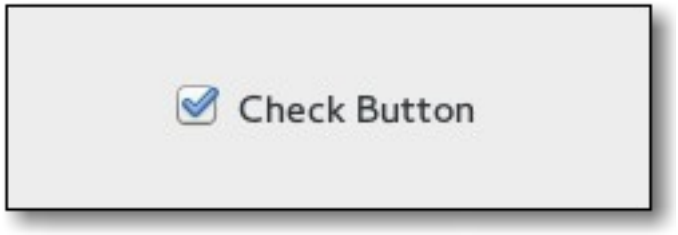
Parameters

button	the object that received the signal
user_data	user data set when the signal handler was connected.

Flags: Run First

GtkCheckButton

GtkCheckButton — Create widgets with a discrete toggle button



Functions

GtkWidget *	gtk_check_button_new ()
GtkWidget *	gtk_check_button_new_with_label ()
GtkWidget *	gtk_check_button_new_with_mnemonic ()

Style Properties

gint	indicator-size	Read
gint	indicator-spacing	Read

Types and Values

struct	GtkCheckButton

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkContainer
            └── GtkBin
                └── GtkButton
                    └── GtkToggleButton
                        └── GtkCheckButton
                            └── GtkRadioButton
```

Implemented Interfaces

GtkCheckButton implements AtkImplementorIface, GtkBuildable, GtkActionable and GtkActivatable.

Includes

```
#include <gtk/gtk.h>
```

Description

A GtkCheckButton places a discrete GtkToggleButton next to a widget, (usually a GtkLabel). See the section on

GtkToggleButton widgets for more information about toggle/check buttons.

The important signal (“toggled”) is also inherited from GtkToggleButton.

CSS nodes

```
checkbutton
├── check
└── <child>
```

A GtkCheckButton with indicator (see `gtk_toggle_button_set_mode()`) has a main CSS node with name checkbutton and a subnode with name check.

```
button.check
├── check
└── <child>
```

A GtkCheckButton without indicator changes the name of its main node to button and adds a .check style class to it. The subnode is invisible in this case.

Functions

gtk_check_button_new ()

```
GtkWidget *
gtk_check_button_new (void);
```

Creates a new GtkCheckButton.

Returns

a GtkWidget.

---

gtk_check_button_new_with_label ()

```
GtkWidget *
gtk_check_button_new_with_label (const gchar *label);
```

Creates a new GtkCheckButton with a GtkLabel to the right of it.

Parameters

label	the text for the check button.

Returns

a GtkWidget.

---

gtk_check_button_new_with_mnemonic ()

```
GtkWidget *
gtk_check_button_new_with_mnemonic (const gchar *label);
```

Creates a new GtkCheckButton containing a label. The label will be created using `gtk_label_new_with_mnemonic()`, so underscores in `label` indicate the mnemonic for the check button.

Parameters

label	The text of the button, with an underscore in front of the mnemonic character

Returns

a new GtkCheckButton

Types and Values

struct GtkCheckButton

```
struct GtkCheckButton;
```

Style Property Details

The `"indicator-size"` style property

```
  "indicator-size"           gint
```

The size of the indicator.

`GtkCheckButton:indicator-size` has been deprecated since version 3.20 and should not be used in newly-written code.

Use CSS min-width and min-height on the indicator node.

Flags: Read

Allowed values: >= 0

Default value: 16

The `"indicator-spacing"` style property

```
  "indicator-spacing"        gint
```

The spacing around the indicator.

`GtkCheckButton:indicator-spacing` has been deprecated since version 3.20 and should not be used in newly-written code.

Use CSS margins of the indicator node, the value of this style property is ignored.

Flags: Read

Allowed values: >= 0

Default value: 2

See Also

GtkCheckMenuItem, GtkButton, GtkToggleButton, GtkRadioButton

GtkRadioButton

GtkRadioButton — A choice from multiple check buttons

Functions

GtkWidget *	gtk_radio_button_new ()
GtkWidget *	gtk_radio_button_new_from_widget ()
GtkWidget *	gtk_radio_button_new_with_label ()
GtkWidget *	gtk_radio_button_new_with_label_from_widget ()
GtkWidget *	gtk_radio_button_new_with_mnemonic ()
GtkWidget *	gtk_radio_button_new_with_mnemonic_from_widget ()
void	gtk_radio_button_set_group ()
GSList *	gtk_radio_button_get_group ()
void	gtk_radio_button_join_group ()

Properties

GtkRadioButton *	group	Write

Signals

void	group-changed	Run First

Types and Values

struct	GtkRadioButton

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkContainer
            └── GtkBin
                └── GtkButton
                    └── GtkToggleButton
                        └── GtkCheckButton
                            └── GtkRadioButton
```

Implemented Interfaces

GtkRadioButton implements AtkImplementorIface, GtkBuildable, GtkActionable and GtkActivatable.

Includes

```
#include <gtk/gtk.h>
```

Description

A single radio button performs the same basic function as a GtkCheckButton, as its position in the object hierarchy reflects. It is only when multiple radio buttons are grouped together that they become a different user interface component in their own right.

Every radio button is a member of some group of radio buttons. When one is selected, all other radio buttons in the same group are deselected. A GtkRadioButton is one way of giving the user a choice from many options.

Radio button widgets are created with `gtk_radio_button_new()`, passing `NULL` as the argument if this is the first radio button in a group. In subsequent calls, the group you wish to add this button to should be passed as an argument. Optionally, `gtk_radio_button_new_with_label()` can be used if you want a text label on the radio button.

Alternatively, when adding widgets to an existing group of radio buttons, use `gtk_radio_button_new_from_widget()` with a GtkRadioButton that already has a group assigned to it. The convenience function `gtk_radio_button_new_with_label_from_widget()` is also provided.

To retrieve the group a GtkRadioButton is assigned to, use `gtk_radio_button_get_group()`.

To remove a GtkRadioButton from one group and make it part of a new one, use `gtk_radio_button_set_group()`.

The group list does not need to be freed, as each GtkRadioButton will remove itself and its list item when it is destroyed.

CSS nodes

```
1                                                   radiobutton
2                                                   ├── radio
3                                                   └── <child>
```

A GtkRadioButton with indicator (see `gtk_toggle_button_set_mode()`) has a main CSS node with name radiobutton and a subnode with name radio.

```
1                                                   button.radio
2                                                   ├── radio
3                                                   └── <child>
```

A GtkRadioButton without indicator changes the name of its main node to button and adds a .radio style class to it. The subnode is invisible in this case.

How to create a group of two radio buttons.

```
1    void create_radio_buttons (void) {
2
3       GtkWidget *window, *radio1, *radio2, *box, *entry;
4       window = gtk_window_new (GTK_WINDOW_TOPLEVEL);
5       box = gtk_box_new (GTK_ORIENTATION_VERTICAL, 2);
6       gtk_box_set_homogeneous (GTK_BOX (box), TRUE);
7
8       // Create a radio button with a GtkEntry widget
9       radio1 = gtk_radio_button_new (NULL);
10      entry = gtk_entry_new ();
11      gtk_container_add (GTK_CONTAINER (radio1), entry);
12
13
14      // Create a radio button with a label
15      radio2 = gtk_radio_button_new_with_label_from_widget (GTK_RADIO_BUTTON (radio1),
16                                                            "I’m the second radio
17   button.");
18
19      // Pack them into a box, then show all the widgets
20      gtk_box_pack_start (GTK_BOX (box), radio1);
21      gtk_box_pack_start (GTK_BOX (box), radio2);
22      gtk_container_add (GTK_CONTAINER (window), box);
23      gtk_widget_show_all (window);
24      return;
     }
```

When an unselected button in the group is clicked the clicked button receives the “toggled” signal, as does the previously selected button. Inside the “toggled” handler, `gtk_toggle_button_get_active()` can be used to determine if the button has been selected or deselected.

Functions

gtk_radio_button_new ()

```
GtkWidget *
gtk_radio_button_new (GSList *group);
```

Creates a new GtkRadioButton. To be of any practical value, a widget should then be packed into the radio button.

Parameters

group	an existing radio button group, or `NULL` if you are creating a new group.	[element-type GtkRadioButton] [allow-none]

Returns

a new radio button

---

gtk_radio_button_new_from_widget ()

```
GtkWidget *
gtk_radio_button_new_from_widget (GtkRadioButton *radio_group_member);
```

Creates a new GtkRadioButton, adding it to the same group as `radio_group_member` . As with `gtk_radio_button_new()`, a widget should be packed into the radio button.

[constructor]

Parameters

radio_group_member	an existing GtkRadioButton.	[allow-none]

Returns

a new radio button.

[transfer none]

---

gtk_radio_button_new_with_label ()

```
GtkWidget *
gtk_radio_button_new_with_label (GSList *group,
                                 const gchar *label);
```

Creates a new GtkRadioButton with a text label.

Parameters

group	an existing radio button group, or `NULL` if you are creating a new group.	[element-type GtkRadioButton] [allow-none]
label	the text label to display next to the radio button.	

Returns

a new radio button.

---

gtk_radio_button_new_with_label_from_widget ()

```
GtkWidget *
gtk_radio_button_new_with_label_from_widget
                                (GtkRadioButton *radio_group_member,
                                 const gchar *label);
```

Creates a new GtkRadioButton with a text label, adding it to the same group as `radio_group_member` .

[constructor]

Parameters

radio_group_member	widget to get radio group from or `NULL`.	[allow-none]
label	a text string to display next to the radio button.	

Returns

a new radio button.

[transfer none]

---

gtk_radio_button_new_with_mnemonic ()

```
GtkWidget *
gtk_radio_button_new_with_mnemonic (GSList *group,
                                    const gchar *label);
```

Creates a new GtkRadioButton containing a label, adding it to the same group as `group` . The label will be created using `gtk_label_new_with_mnemonic()`, so underscores in `label` indicate the mnemonic for the button.

Parameters

group	the radio button group, or `NULL`.	[element-type GtkRadioButton] [allow-none]
label	the text of the button, with an underscore in front of the mnemonic character	

Returns

a new GtkRadioButton

---

gtk_radio_button_new_with_mnemonic_from_widget ()

```
GtkWidget *
gtk_radio_button_new_with_mnemonic_from_widget
                                (GtkRadioButton *radio_group_member,
                                 const gchar *label);
```

Creates a new GtkRadioButton containing a label. The label will be created using `gtk_label_new_with_mnemonic()`, so underscores in `label` indicate the mnemonic for the button.

[constructor]

Parameters

radio_group_member	widget to get radio group from or `NULL`.	[allow-none]
label	the text of the button, with an underscore in front of the mnemonic character	

Returns

a new GtkRadioButton.

[transfer none]

---

gtk_radio_button_set_group ()

```
void
gtk_radio_button_set_group (GtkRadioButton *radio_button,
                            GSList *group);
```

Sets a GtkRadioButton's group. It should be noted that this does not change the layout of your interface in any way, so if you are changing the group, it is likely you will need to re-arrange the user interface to reflect these changes.

Parameters

radio_button	a GtkRadioButton.	
group	an existing radio button group, such as one returned from `gtk_radio_button_get_group()`, or `NULL`.	[element-type GtkRadioButton] [allow-none]

---

gtk_radio_button_get_group ()

```
GSList *
gtk_radio_button_get_group (GtkRadioButton *radio_button);
```

Retrieves the group assigned to a radio button.

Parameters

radio_button	a GtkRadioButton.

Returns

a linked list containing all the radio buttons in the same group as `radio_button` . The returned list is owned by the radio button and must not be modified or freed.

[element-type GtkRadioButton][transfer none]

---

gtk_radio_button_join_group ()

```
void
gtk_radio_button_join_group (GtkRadioButton *radio_button,
                             GtkRadioButton *group_source);
```

Joins a GtkRadioButton object to the group of another GtkRadioButton object

Use this in language bindings instead of the `gtk_radio_button_get_group()` and `gtk_radio_button_set_group()` methods

A common way to set up a group of radio buttons is the following:

```
1   GtkRadioButton *radio_button;
2   GtkRadioButton *last_button;
3
4   while (some_condition)
5     {
6        radio_button = gtk_radio_button_new (NULL);
7
8        gtk_radio_button_join_group (radio_button, last_button);
9        last_button = radio_button;
10    }
```

Parameters

radio_button	the GtkRadioButton object	
group_source	a radio button object whos group we are joining, or `NULL` to remove the radio button from its group.	[allow-none]

Since: 3.0

Types and Values

struct GtkRadioButton

```
struct GtkRadioButton;
```

Property Details

The "group" property

```
  "group"                    GtkRadioButton *
```

Sets a new group for a radio button.

Flags: Write

Signal Details

The "group-changed" signal

```
void
user_function (GtkRadioButton *button,
               gpointer        user_data)
```

Emitted when the group of radio buttons that a radio button belongs to changes. This is emitted when a radio button switches from being alone to being part of a group of 2 or more buttons, or vice-versa, and when a button is moved from one group of 2 or more buttons to a different one, but not when the composition of the group that a button belongs to changes.

Parameters

button	the object which received the signal
user_data	user data set when the signal handler was connected.

Flags: Run First

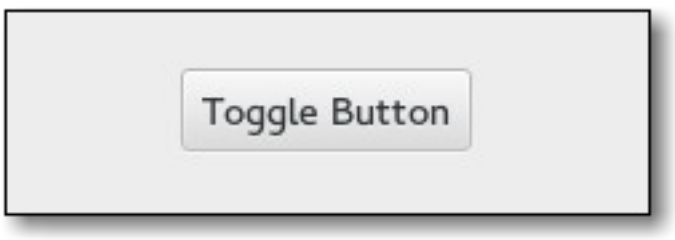
Since: 2.4

See Also

GtkComboBox

GtkToggleButton

GtkToggleButton — Create buttons which retain their state



Functions

GtkWidget *	gtk_toggle_button_new ()
GtkWidget *	gtk_toggle_button_new_with_label ()

GtkWidget *	gtk_toggle_button_new_with_mnemonic ()
void	gtk_toggle_button_set_mode ()
gboolean	gtk_toggle_button_get_mode ()
void	gtk_toggle_button_toggled ()
gboolean	gtk_toggle_button_get_active ()
void	gtk_toggle_button_set_active ()
gboolean	gtk_toggle_button_get_inconsistent ()
void	gtk_toggle_button_set_inconsistent ()

Properties

gboolean	active	Read / Write
gboolean	draw-indicator	Read / Write
gboolean	inconsistent	Read / Write

Signals

void	toggled	Run First

Types and Values

struct	GtkToggleButton

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkContainer
            └── GtkBin
                └── GtkButton
                    └── GtkToggleButton
                        ├── GtkCheckButton
                        └── GtkMenuButton
```

Implemented Interfaces

GtkToggleButton implements AtkImplementorIface, GtkBuildable, GtkActionable and GtkActivatable.

Includes

```
#include <gtk/gtk.h>
```

Description

A GtkToggleButton is a GtkButton which will remain “pressed-in” when clicked. Clicking again will cause the toggle button to return to its normal state.

A toggle button is created by calling either `gtk_toggle_button_new()` or `gtk_toggle_button_new_with_label()`. If using the former, it is advisable to pack a widget, (such as a

GtkLabel and/or a GtkImage), into the toggle button's container. (See GtkButton for more information).

The state of a GtkToggleButton can be set specifically using `gtk_toggle_button_set_active()`, and retrieved using `gtk_toggle_button_get_active()`.

To simply switch the state of a toggle button, use `gtk_toggle_button_toggled()`.

CSS nodes

GtkToggleButton has a single CSS node with name button. To differentiate it from a plain GtkButton, it gets the .toggle style class.

Creating two GtkToggleButton widgets.

```
1   static void output_state (GtkToggleButton *source, gpointer user_data) {
2     printf ("Active: %d\n", gtk_toggle_button_get_active (source));
3   }
4
5   void make_toggles (void) {
6     GtkWidget *window, *toggle1, *toggle2;
7     GtkWidget *box;
8     const char *text;
9
10    window = gtk_window_new (GTK_WINDOW_TOPLEVEL);
11    box = gtk_box_new (GTK_ORIENTATION_VERTICAL, 12);
12
13    text = "Hi, I'm a toggle button.";
14    toggle1 = gtk_toggle_button_new_with_label (text);
15
16    // Makes this toggle button invisible
17    gtk_toggle_button_set_mode (GTK_TOGGLE_BUTTON (toggle1),
18                                TRUE);
19
20    g_signal_connect (toggle1, "toggled",
21                      G_CALLBACK (output_state),
22                      NULL);
23    gtk_container_add (GTK_CONTAINER (box), toggle1);
24
25    text = "Hi, I'm a toggle button.";
26    toggle2 = gtk_toggle_button_new_with_label (text);
27    gtk_toggle_button_set_mode (GTK_TOGGLE_BUTTON (toggle2),
28                                FALSE);
29    g_signal_connect (toggle2, "toggled",
30                      G_CALLBACK (output_state),
31                      NULL);
32    gtk_container_add (GTK_CONTAINER (box), toggle2);
33
34    gtk_container_add (GTK_CONTAINER (window), box);
35    gtk_widget_show_all (window);
36  }
```

Functions

gtk_toggle_button_new ()

```
GtkWidget *
gtk_toggle_button_new (void);
```

Creates a new toggle button. A widget should be packed into the button, as in `gtk_button_new()`.

Returns

a new toggle button.

---

gtk_toggle_button_new_with_label ()

```
GtkWidget *
gtk_toggle_button_new_with_label (const gchar *label);
```

Creates a new toggle button with a text label.

Parameters

label	a string containing the message to be placed in the toggle button.

Returns

a new toggle button.

---

gtk_toggle_button_new_with_mnemonic ()

```
GtkWidget *
gtk_toggle_button_new_with_mnemonic (const gchar *label);
```

Creates a new GtkToggleButton containing a label. The label will be created using `gtk_label_new_with_mnemonic()`, so underscores in `label` indicate the mnemonic for the button.

Parameters

label	the text of the button, with an underscore in front of the mnemonic character

Returns

a new GtkToggleButton

---

gtk_toggle_button_set_mode ()

```
void
gtk_toggle_button_set_mode (GtkToggleButton *toggle_button,
                            gboolean draw_indicator);
```

Sets whether the button is displayed as a separate indicator and label. You can call this function on a checkbutton or a radiobutton with `draw_indicator` = `FALSE` to make the button look like a normal button.

This can be used to create linked strip of buttons that work like a GtkStackSwitcher.

This function only affects instances of classes like GtkCheckButton and GtkRadioButton that derive from GtkToggleButton, not instances of GtkToggleButton itself.

Parameters

toggle_button	a GtkToggleButton
draw_indicator	if `TRUE`, draw the button as a separate indicator and label; if `FALSE`, draw the button like a normal button

---

gtk_toggle_button_get_mode ()

```
gboolean
gtk_toggle_button_get_mode (GtkToggleButton *toggle_button);
```

Retrieves whether the button is displayed as a separate indicator and label. See `gtk_toggle_button_set_mode()`.

Parameters

toggle_button	a GtkToggleButton

Returns

`TRUE` if the togglebutton is drawn as a separate indicator and label.

---

gtk_toggle_button_toggled ()

```
void
gtk_toggle_button_toggled (GtkToggleButton *toggle_button);
```

Emits the “toggled” signal on the GtkToggleButton. There is no good reason for an application ever to call this function.

Parameters

toggle_button	a GtkToggleButton.

---

gtk_toggle_button_get_active ()

```
gboolean
gtk_toggle_button_get_active (GtkToggleButton *toggle_button);
```

Queries a GtkToggleButton and returns its current state. Returns TRUE if the toggle button is pressed in and FALSE if it is raised.

Parameters

toggle_button	a GtkToggleButton.

Returns

a gboolean value.

---

gtk_toggle_button_set_active ()

```
void
gtk_toggle_button_set_active (GtkToggleButton *toggle_button,
                              gboolean is_active);
```

Sets the status of the toggle button. Set to TRUE if you want the GtkToggleButton to be “pressed in”, and FALSE to raise it. This action causes the “toggled” signal and the “clicked” signal to be emitted.

Parameters

toggle_button	a GtkToggleButton.
is_active	TRUE or FALSE.

---

gtk_toggle_button_get_inconsistent ()

```
gboolean
gtk_toggle_button_get_inconsistent (GtkToggleButton *toggle_button);
```

Gets the value set by gtk_toggle_button_set_inconsistent().

Parameters

toggle_button	a GtkToggleButton

Returns

TRUE if the button is displayed as inconsistent, FALSE otherwise

---

gtk_toggle_button_set_inconsistent ()

```
void
gtk_toggle_button_set_inconsistent (GtkToggleButton *toggle_button,
                                    gboolean setting);
```

If the user has selected a range of elements (such as some text or spreadsheet cells) that are affected by a toggle button, and the current values in that range are inconsistent, you may want to display the toggle in an “in between” state. This function turns on “in between” display. Normally you would turn off the inconsistent state again if the user toggles the toggle button. This has to be done manually, `gtk_toggle_button_set_inconsistent()` only affects visual appearance, it doesn’t affect the semantics of the button.

Parameters

toggle_button	a GtkToggleButton
setting	`TRUE` if state is inconsistent

Types and Values

struct GtkToggleButton

```
struct GtkToggleButton;
```

Property Details

The “active” property

“active” gboolean

If the toggle button should be pressed in.

Flags: Read / Write

Default value: FALSE

The “draw-indicator” property

“draw-indicator” gboolean

If the toggle part of the button is displayed.

Flags: Read / Write

Default value: FALSE

The “inconsistent” property

“inconsistent” gboolean

If the toggle button is in an "in between" state.

Flags: Read / Write

Default value: FALSE

Signal Details

The `"toggled"` signal

```
void
user_function (GtkToggleButton *togglebutton,
               gpointer         user_data)
```

Should be connected if you wish to perform an action whenever the GtkToggleButton's state is changed.

Parameters

togglebutton	the object which received the signal.
user_data	user data set when the signal handler was connected.

Flags: Run First

See Also

GtkButton, GtkCheckButton, GtkCheckMenuItem

GtkLinkButton

GtkLinkButton — Create buttons bound to a URL

Functions

GtkWidget *	gtk_link_button_new ()
GtkWidget *	gtk_link_button_new_with_label ()
const gchar *	gtk_link_button_get_uri ()
void	gtk_link_button_set_uri ()
gboolean	gtk_link_button_get_visited ()
void	gtk_link_button_set_visited ()

Properties

gchar *	uri	Read / Write
gboolean	visited	Read / Write

Signals

gboolean	activate-link	Run Last

Types and Values

struct	GtkLinkButton
struct	GtkLinkButtonClass

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkContainer
            └── GtkBin
                └── GtkButton
                    └── GtkLinkButton
```

Implemented Interfaces

GtkLinkButton implements AtkImplementorIface, GtkBuildable, GtkActionable and GtkActivatable.

Includes

```
#include <gtk/gtk.h>
```

Description

A GtkLinkButton is a GtkButton with a hyperlink, similar to the one used by web browsers, which triggers an action when clicked. It is useful to show quick links to resources.

A link button is created by calling either `gtk_link_button_new()` or `gtk_link_button_new_with_label()`. If using the former, the URI you pass to the constructor is used as a label for the widget.

The URI bound to a GtkLinkButton can be set specifically using `gtk_link_button_set_uri()`, and retrieved using `gtk_link_button_get_uri()`.

By default, GtkLinkButton calls `gtk_show_uri_on_window()` when the button is clicked. This behaviour can be overridden by connecting to the "activate-link" signal and returning `TRUE` from the signal handler.

CSS nodes

GtkLinkButton has a single CSS node with name button. To differentiate it from a plain GtkButton, it gets the .link style class.

Functions

gtk_link_button_new ()

```
GtkWidget *
gtk_link_button_new (const gchar *uri);
```

Creates a new GtkLinkButton with the URI as its text.

Parameters

uri	a valid URI	

Returns

a new link button widget.

Since: 2.10

---

gtk_link_button_new_with_label ()

```
GtkWidget *
gtk_link_button_new_with_label (const gchar *uri,
                                const gchar *label);
```

Creates a new GtkLinkButton containing a label.

Parameters

uri	a valid URI	
label	the text of the button.	[allow-none]

Returns

a new link button widget.

[transfer none]

Since: 2.10

---

gtk_link_button_get_uri ()

```
const gchar *
gtk_link_button_get_uri (GtkLinkButton *link_button);
```

Retrieves the URI set using `gtk_link_button_set_uri()`.

Parameters

link_button	a GtkLinkButton

Returns

a valid URI. The returned string is owned by the link button and should not be modified or freed.

Since: 2.10

---

gtk_link_button_set_uri ()

```
void
gtk_link_button_set_uri (GtkLinkButton *link_button,
                         const gchar *uri);
```

Sets `uri` as the URI where the GtkLinkButton points. As a side-effect this unsets the “visited” state of the button.

Parameters

link_button	a GtkLinkButton
uri	a valid URI

Since: 2.10

---

gtk_link_button_get_visited ()

```
gboolean
gtk_link_button_get_visited (GtkLinkButton *link_button);
```

Retrieves the “visited” state of the URI where the GtkLinkButton points. The button becomes visited when it is clicked. If the URI is changed on the button, the “visited” state is unset again.

The state may also be changed using `gtk_link_button_set_visited()`.

Parameters

link_button	a GtkLinkButton

Returns

`TRUE` if the link has been visited, `FALSE` otherwise

Since: 2.14

---

gtk_link_button_set_visited ()

```
void
gtk_link_button_set_visited (GtkLinkButton *link_button,
                             gboolean visited);
```

Sets the “visited” state of the URI where the GtkLinkButton points. See `gtk_link_button_get_visited()` for more details.

Parameters

link_button	a GtkLinkButton
visited	the new “visited” state

Since: 2.14

Types and Values

struct GtkLinkButton

```
struct GtkLinkButton;
```

The GtkLinkButton contains only private data and should be accessed using the provided API.

---

struct GtkLinkButtonClass

```
struct GtkLinkButtonClass {
  gboolean (* activate_link) (GtkLinkButton *button);
};
```

The GtkLinkButtonClass contains only private data.

Members

`activate_link` ()	class handler for the “activate-link” signal

Property Details

The `“uri”` property

```
  “uri”                      gchar *
```

The URI bound to this button.

Flags: Read / Write

Default value: NULL

Since: 2.10

The “visited” property

```
  “visited”                  gboolean
```

The 'visited' state of this button. A visited link is drawn in a different color.

Flags: Read / Write

Default value: FALSE

Since: 2.14

Signal Details

The “activate-link” signal

```
gboolean
user_function (GtkLinkButton *button,
               gpointer       user_data)
```

The ::activate-link signal is emitted each time the GtkLinkButton has been clicked.

The default handler will call `gtk_show_uri_on_window()` with the URI stored inside the “uri” property.

To override the default behavior, you can connect to the ::activate-link signal and stop the propagation of the signal by returning `TRUE` from your handler.

Parameters

button	the GtkLinkButton that emitted the signal
user_data	user data set when the signal handler was connected.

Flags: Run Last

See Also

GtkButton

GtkMenuButton

GtkMenuButton — A widget that shows a popup when clicked on

Functions

GtkWidget *	gtk_menu_button_new ()
void	gtk_menu_button_set_popup ()
GtkMenu *	gtk_menu_button_get_popup ()
void	gtk_menu_button_set_popover ()
GtkPopover *	gtk_menu_button_get_popover ()
void	gtk_menu_button_set_menu_model ()
GMenuModel *	gtk_menu_button_get_menu_model ()
void	gtk_menu_button_set_use_popover ()
gboolean	gtk_menu_button_get_use_popover ()
void	gtk_menu_button_set_direction ()
GtkArrowType	gtk_menu_button_get_direction ()
void	gtk_menu_button_set_align_widget ()
GtkWidget *	gtk_menu_button_get_align_widget ()

Properties

GtkContainer *	align-widget	Read / Write
GtkArrowType	direction	Read / Write
GMenuModel *	menu-model	Read / Write
GtkPopover *	popover	Read / Write
GtkMenu *	popup	Read / Write
gboolean	use-popover	Read / Write

Types and Values

struct	GtkMenuButton
enum	GtkArrowType

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkContainer
            └── GtkBin
                └── GtkButton
                    └── GtkToggleButton
                        └── GtkMenuButton
```

Implemented Interfaces

GtkMenuButton implements AtkImplementorIface, GtkBuildable, GtkActionable and GtkActivatable.

Includes

```
#include <gtk/gtk.h>
```

Description

The GtkMenuButton widget is used to display a popup when clicked on. This popup can be provided either as a GtkMenu, a GtkPopover or an abstract GMenuModel.

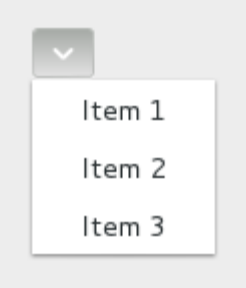
The GtkMenuButton widget can hold any valid child widget. That is, it can hold almost any other standard GtkWidget. The most commonly used child is GtkImage. If no widget is explicitely added to the GtkMenuButton, a GtkImage is automatically created, using an arrow image oriented according to “direction” or the generic “open-menu-symbolic” icon if the direction is not set.

The positioning of the popup is determined by the “direction” property of the menu button.

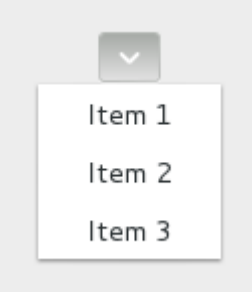
For menus, the “halign” and “valign” properties of the menu are also taken into account. For example, when the direction is `GTK_ARROW_DOWN` and the horizontal alignment is `GTK_ALIGN_START`, the menu will be positioned below the button, with the starting edge (depending on the text direction) of the menu aligned with the starting edge of the button. If there is not enough space below the button, the menu is popped up above the button instead. If the alignment would move part of the menu offscreen, it is “pushed in”.

Direction = Down

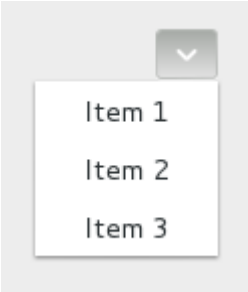
- halign = start



- halign = center



- halign = end



Direction = Up

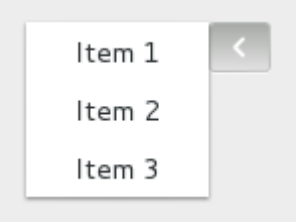
- halign = start

- halign = center

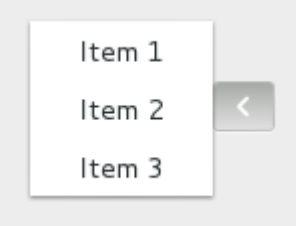
- halign = end

Direction = Left

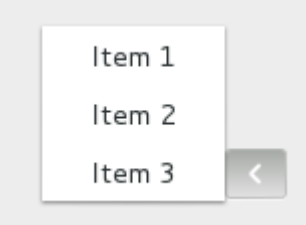
- valign = start



- valign = center

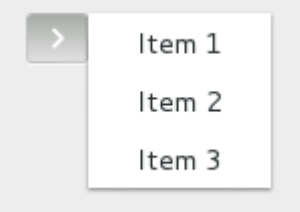


- valign = end

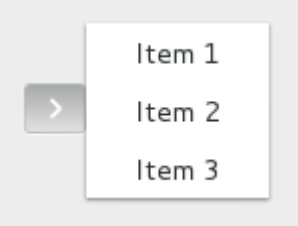


Direction = Right

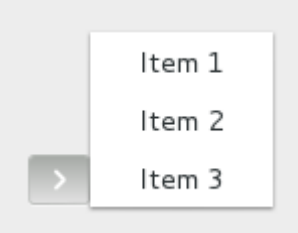
- valign = start



- valign = center



- valign = end



CSS nodes

GtkMenuButton has a single CSS node with name button. To differentiate it from a plain GtkButton, it gets the .popup style class.

Functions

gtk_menu_button_new ()

```
GtkWidget *
gtk_menu_button_new (void);
```

Creates a new GtkMenuButton widget with downwards-pointing arrow as the only child. You can replace the child widget with another GtkWidget should you wish to.

Returns

The newly created GtkMenuButton widget

Since: 3.6

---

gtk_menu_button_set_popup ()

```
void
gtk_menu_button_set_popup (GtkMenuButton *menu_button,
                           GtkWidget *menu);
```

Sets the GtkMenu that will be popped up when the `menu_button` is clicked, or `NULL` to dissociate any existing menu and disable the button.

If “menu-model” or “popover” are set, those objects are dissociated from the `menu_button` , and those properties are set to `NULL`.

Parameters

menu_button	a GtkMenuButton	
menu	a GtkMenu, or `NULL` to unset and disable the button.	[nullable]

Since: 3.6

---

gtk_menu_button_get_popup ()

```
GtkMenu *
gtk_menu_button_get_popup (GtkMenuButton *menu_button);
```

Returns the GtkMenu that pops out of the button. If the button does not use a GtkMenu, this function returns

NULL.

Parameters

menu_button	a GtkMenuButton

Returns

a GtkMenu or NULL.

[nullable][transfer none]

Since: 3.6

---

gtk_menu_button_set_popover ()

```
void
gtk_menu_button_set_popover (GtkMenuButton *menu_button,
                             GtkWidget *popover);
```

Sets the GtkPopover that will be popped up when the menu_button is clicked, or NULL to dissociate any existing popover and disable the button.

If “menu-model” or “popup” are set, those objects are dissociated from the menu_button , and those properties are set to NULL.

Parameters

menu_button	a GtkMenuButton
popover	a GtkPopover, or NULL to unset and disable the button. [nullable]

Since: 3.12

---

gtk_menu_button_get_popover ()

```
GtkPopover *
gtk_menu_button_get_popover (GtkMenuButton *menu_button);
```

Returns the GtkPopover that pops out of the button. If the button is not using a GtkPopover, this function returns NULL.

Parameters

menu_button	a GtkMenuButton

Returns

a GtkPopover or NULL.

[nullable][transfer none]

Since: 3.12

---

gtk_menu_button_set_menu_model ()

```
void
gtk_menu_button_set_menu_model (GtkMenuButton *menu_button,
                                GMenuModel *menu_model);
```

Sets the GMenuModel from which the popup will be constructed, or `NULL` to dissociate any existing menu model and disable the button.

Depending on the value of "use-popover", either a GtkMenu will be created with `gtk_menu_new_from_model()`, or a GtkPopover with `gtk_popover_new_from_model()`. In either case, actions will be connected as documented for these functions.

If "popup" or "popover" are already set, those widgets are dissociated from the `menu_button` , and those properties are set to `NULL`.

Parameters

menu_button	a GtkMenuButton	
menu_model	a GMenuModel, or `NULL` to unset and disable the button.	[nullable]

Since: 3.6

---

gtk_menu_button_get_menu_model ()

```
GMenuModel *
gtk_menu_button_get_menu_model (GtkMenuButton *menu_button);
```

Returns the GMenuModel used to generate the popup.

Parameters

menu_button	a GtkMenuButton

Returns

a GMenuModel or `NULL`.

[nullable][transfer none]

Since: 3.6

---

gtk_menu_button_set_use_popover ()

```
void
gtk_menu_button_set_use_popover (GtkMenuButton *menu_button,
                                 gboolean use_popover);
```

Sets whether to construct a GtkPopover instead of GtkMenu when `gtk_menu_button_set_menu_model()` is called. Note that this property is only consulted when a new menu model is set.

Parameters

menu_button	a GtkMenuButton
use_popover	`TRUE` to construct a popover from the menu model

Since: 3.12

---

gtk_menu_button_get_use_popover ()

```
gboolean
gtk_menu_button_get_use_popover (GtkMenuButton *menu_button);
```

Returns whether a GtkPopover or a GtkMenu will be constructed from the menu model.

Parameters

menu_button	a GtkMenuButton

Returns

`TRUE` if using a GtkPopover

Since: 3.12

---

gtk_menu_button_set_direction ()

```
void
gtk_menu_button_set_direction (GtkMenuButton *menu_button,
                               GtkArrowType direction);
```

Sets the direction in which the popup will be popped up, as well as changing the arrow's direction. The child will not be changed to an arrow if it was customized.

If the does not fit in the available space in the given direction, GTK+ will its best to keep it inside the screen and fully visible.

If you pass `GTK_ARROW_NONE` for a `direction` , the popup will behave as if you passed `GTK_ARROW_DOWN` (although you won't see any arrows).

Parameters

menu_button	a GtkMenuButton	
direction	a GtkArrowType	

Since: 3.6

---

gtk_menu_button_get_direction ()

```
GtkArrowType
gtk_menu_button_get_direction (GtkMenuButton *menu_button);
```

Returns the direction the popup will be pointing at when popped up.

Parameters

menu_button	a GtkMenuButton	

Returns

a GtkArrowType value

Since: 3.6

---

gtk_menu_button_set_align_widget ()

```
void
gtk_menu_button_set_align_widget (GtkMenuButton *menu_button,
                                  GtkWidget *align_widget);
```

Sets the GtkWidget to use to line the menu with when popped up. Note that the `align_widget` must contain the GtkMenuButton itself.

Setting it to `NULL` means that the menu will be aligned with the button itself.

Note that this property is only used with menus currently, and not for popovers.

Parameters

menu_button	a GtkMenuButton	
align_widget	a GtkWidget.	[allow-none]

Since: 3.6

---

gtk_menu_button_get_align_widget ()

```
GtkWidget *
gtk_menu_button_get_align_widget (GtkMenuButton *menu_button);
```

Returns the parent GtkWidget to use to line up with menu.

Parameters

menu_button	a GtkMenuButton

Returns

a GtkWidget value or `NULL`.

[nullable][transfer none]

Since: 3.6

Types and Values

struct GtkMenuButton

```
struct GtkMenuButton;
```

---

enum GtkArrowType

Used to indicate the direction in which an arrow should point.

Members

GTK_ARROW_UP	Represents an upward pointing arrow.
GTK_ARROW_DOWN	Represents a downward pointing arrow.
GTK_ARROW_LEFT	Represents a left pointing arrow.
GTK_ARROW_RIGHT	Represents a right pointing arrow.
GTK_ARROW_NONE	No arrow. Since 2.10.

Property Details

The `“align-widget”` property

```
  “align-widget”             GtkContainer *
```

The GtkWidget to use to align the menu with.

Flags: Read / Write

Since: 3.6

The “direction” property

“direction” GtkArrowType

The GtkArrowType representing the direction in which the menu or popover will be popped out.

Flags: Read / Write

Default value: GTK_ARROW_DOWN

Since: 3.6

The “menu-model” property

“menu-model” GMenuModel *

The GMenuModel from which the popup will be created. Depending on the “use-popover” property, that may be a menu or a popover.

See gtk_menu_button_set_menu_model() for the interaction with the “popup” property.

Flags: Read / Write

Since: 3.6

The “popover” property

“popover” GtkPopover *

The GtkPopover that will be popped up when the button is clicked.

Flags: Read / Write

Since: 3.12

The “popup” property

“popup” GtkMenu *

The GtkMenu that will be popped up when the button is clicked.

Flags: Read / Write

Since: 3.6

The “use-popover” property

“use-popover” gboolean

Whether to construct a GtkPopover from the menu model, or a GtkMenu.

Flags: Read / Write

Default value: TRUE

Since: 3.12

GtkSwitch

GtkSwitch — A “light switch” style toggle

Functions

GtkWidget *	gtk_switch_new ()
void	gtk_switch_set_active ()
gboolean	gtk_switch_get_active ()
void	gtk_switch_set_state ()
gboolean	gtk_switch_get_state ()

Properties

gboolean	active	Read / Write
gboolean	state	Read / Write

Style Properties

gint	slider-height	Read
gint	slider-width	Read

Signals

void	activate	Action
gboolean	state-set	Run Last

Types and Values

struct	GtkSwitch
struct	GtkSwitchClass

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkSwitch
```

Implemented Interfaces

GtkSwitch implements AtkImplementorIface, GtkBuildable, GtkActionable and GtkActivatable.

Includes

```
#include <gtk/gtk.h>
```

Description

GtkSwitch is a widget that has two states: on or off. The user can control which state should be active by clicking the empty area, or by dragging the handle.

GtkSwitch can also handle situations where the underlying state changes with a delay. See “state-set” for details.

CSS nodes

```
1                                                    switch
2                                                    └── slider
```

GtkSwitch has two css nodes, the main node with the name switch and a subnode named slider. Neither of them is using any style classes.

Functions

gtk_switch_new ()

```
GtkWidget *
gtk_switch_new (void);
```

Creates a new GtkSwitch widget.

Returns

the newly created GtkSwitch instance

Since: 3.0

---

gtk_switch_set_active ()

```
void
gtk_switch_set_active (GtkSwitch *sw,
                       gboolean is_active);
```

Changes the state of `sw` to the desired one.

Parameters

sw	a GtkSwitch
is_active	TRUE if sw should be active, and FALSE otherwise

Since: 3.0

---

gtk_switch_get_active ()

```
gboolean
gtk_switch_get_active (GtkSwitch *sw);
```

Gets whether the GtkSwitch is in its “on” or “off” state.

Parameters

sw	a GtkSwitch

Returns

TRUE if the GtkSwitch is active, and FALSE otherwise

Since: 3.0

---

gtk_switch_set_state ()

```
void
gtk_switch_set_state (GtkSwitch *sw,
                      gboolean state);
```

Sets the underlying state of the GtkSwitch.

Normally, this is the same as “active”, unless the switch is set up for delayed state changes. This function is typically called from a “state-set” signal handler.

See “state-set” for details.

Parameters

sw	a GtkSwitch
state	the new state

Since: 3.14

---

gtk_switch_get_state ()

```
gboolean
gtk_switch_get_state (GtkSwitch *sw);
```

Gets the underlying state of the GtkSwitch.

Parameters

sw	a GtkSwitch

Returns

the underlying state

Since: 3.14

Types and Values

struct GtkSwitch

```
struct GtkSwitch;
```

The GtkSwitch contains private data and it should only be accessed using the provided API.

---

struct GtkSwitchClass

```
struct GtkSwitchClass {
  GtkWidgetClass parent_class;


  void (* activate) (GtkSwitch *sw);

  gboolean (* state_set) (GtkSwitch *sw, gboolean state);
};
```

Members

`activate` ()	An action signal and emitting it causes the switch to animate.
`state_set` ()	Class handler for the ::state-set signal.

Property Details

The `"active"` property

```
  "active"                   gboolean
```

Whether the GtkSwitch widget is in its on or off state.

Flags: Read / Write

Default value: FALSE

The “state” property

“state” gboolean

The backend state that is controlled by the switch. See “state-set” for details.

Flags: Read / Write

Default value: FALSE

Since: 3.14

Style Property Details

The “slider-height” style property

“slider-height” gint

The minimum height of the GtkSwitch handle, in pixels.

`GtkSwitch:slider-height` has been deprecated since version 3.20 and should not be used in newly-written code.

Use the CSS min-height property instead.

Flags: Read

Allowed values: >= 22

Default value: 22

Since: 3.18

The “slider-width” style property

“slider-width” gint

The minimum width of the GtkSwitch handle, in pixels.

`GtkSwitch:slider-width` has been deprecated since version 3.20 and should not be used in newly-written code.

Use the CSS min-width property instead.

Flags: Read

Allowed values: >= 36

Default value: 36

Signal Details

The “activate” signal

```
void
user_function (GtkSwitch *widget,
               gpointer   user_data)
```

The ::activate signal on GtkSwitch is an action signal and emitting it causes the switch to animate. Applications should never connect to this signal, but use the notify::active signal.

Parameters

widget	the object which received the signal.
user_data	user data set when the signal handler was connected.

Flags: Action

The “state-set” signal

```
gboolean
user_function (GtkSwitch *widget,
               gboolean   state,
               gpointer   user_data)
```

The ::state-set signal on GtkSwitch is emitted to change the underlying state. It is emitted when the user changes the switch position. The default handler keeps the state in sync with the “active” property.

To implement delayed state change, applications can connect to this signal, initiate the change of the underlying state, and call `gtk_switch_set_state()` when the underlying state change is complete. The signal handler should return `TRUE` to prevent the default handler from running.

Visually, the underlying state is represented by the trough color of the switch, while the “active” property is represented by the position of the switch.

Parameters

widget	the object on which the signal was emitted
state	the new state of the switch
user_data	user data set when the signal handler was connected.

Returns

`TRUE` to stop the signal emission

Flags: Run Last

Since: 3.14

See Also

GtkToggleButton

GtkScaleButton

GtkScaleButton — A button which pops up a scale

Functions

GtkWidget *	gtk_scale_button_new ()
void	gtk_scale_button_set_adjustment ()
void	gtk_scale_button_set_icons ()
void	gtk_scale_button_set_value ()
GtkAdjustment *	gtk_scale_button_get_adjustment ()
gdouble	gtk_scale_button_get_value ()
GtkWidget *	gtk_scale_button_get_popup ()
GtkWidget *	gtk_scale_button_get_plus_button ()
GtkWidget *	gtk_scale_button_get_minus_button ()

Properties

GtkAdjustment *	adjustment	Read / Write
GStrv	icons	Read / Write
GtkIconSize	size	Read / Write
gdouble	value	Read / Write

Signals

void	popdown	Action
void	popup	Action
void	value-changed	Run Last

Types and Values

struct	GtkScaleButton

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkContainer
            └── GtkBin
                └── GtkButton
                    └── GtkScaleButton
                        └── GtkVolumeButton
```

Implemented Interfaces

GtkScaleButton implements AtkImplementorIface, GtkBuildable, GtkActionable, GtkActivatable and GtkOrientable.

Includes

```
#include <gtk/gtk.h>
```

Description

GtkScaleButton provides a button which pops up a scale widget. This kind of widget is commonly used for volume controls in multimedia applications, and GTK+ provides a GtkVolumeButton subclass that is tailored for this use case.

CSS nodes

GtkScaleButton has a single CSS node with name button. To differentiate it from a plain GtkButton, it gets the .scale style class.

Functions

gtk_scale_button_new ()

```
GtkWidget *
gtk_scale_button_new (GtkIconSize size,
                      gdouble min,
                      gdouble max,
                      gdouble step,
                      const gchar **icons);
```

Creates a GtkScaleButton, with a range between `min` and `max` , with a stepping of `step` .

Parameters

size	a stock icon size (GtkIconSize).	[type int]
min	the minimum value of the scale (usually 0)	
max	the maximum value of the scale (usually 100)	
step	the stepping of value when a scroll-wheel event, or up/down arrow event occurs (usually 2)	
icons	a `NULL`-terminated array of icon names, or `NULL` if you want to set the list later with `gtk_scale_button_set_icons()`.	[allow-none][array zero-terminated=1]

Returns

a new GtkScaleButton

Since: 2.12

---

gtk_scale_button_set_adjustment ()

```
void
gtk_scale_button_set_adjustment (GtkScaleButton *button,
                                 GtkAdjustment *adjustment);
```

Sets the GtkAdjustment to be used as a model for the GtkScaleButton's scale. See `gtk_range_set_adjustment()` for details.

Parameters

button	a GtkScaleButton	
adjustment	a GtkAdjustment	

Since: 2.12

---

gtk_scale_button_set_icons ()

```
void
gtk_scale_button_set_icons (GtkScaleButton *button,
                            const gchar **icons);
```

Sets the icons to be used by the scale button. For details, see the "icons" property.

Parameters

button	a GtkScaleButton	
icons	a `NULL`-terminated array of icon names.	[array zero-terminated=1]

Since: 2.12

---

gtk_scale_button_set_value ()

```
void
gtk_scale_button_set_value (GtkScaleButton *button,
                            gdouble value);
```

Sets the current value of the scale; if the value is outside the minimum or maximum range values, it will be clamped to fit inside them. The scale button emits the "value-changed" signal if the value changes.

Parameters

button	a GtkScaleButton	

value	new value of the scale button

Since: 2.12

---

gtk_scale_button_get_adjustment ()

```
GtkAdjustment *
gtk_scale_button_get_adjustment (GtkScaleButton *button);
```

Gets the GtkAdjustment associated with the GtkScaleButton's scale. See `gtk_range_get_adjustment()` for details.

Parameters

button	a GtkScaleButton

Returns

the adjustment associated with the scale.

[transfer none]

Since: 2.12

---

gtk_scale_button_get_value ()

```
gdouble
gtk_scale_button_get_value (GtkScaleButton *button);
```

Gets the current value of the scale button.

Parameters

button	a GtkScaleButton

Returns

current value of the scale button

Since: 2.12

---

gtk_scale_button_get_popup ()

```
GtkWidget *
gtk_scale_button_get_popup (GtkScaleButton *button);
```

Retrieves the popup of the GtkScaleButton.

Parameters

button	a GtkScaleButton

Returns

the popup of the GtkScaleButton.

[transfer none]

Since: 2.14

---

gtk_scale_button_get_plus_button ()

```
GtkWidget *
gtk_scale_button_get_plus_button (GtkScaleButton *button);
```

Retrieves the plus button of the GtkScaleButton.

Parameters

button	a GtkScaleButton

Returns

the plus button of the GtkScaleButton as a GtkButton.

[transfer none][type Gtk.Button]

Since: 2.14

---

gtk_scale_button_get_minus_button ()

```
GtkWidget *
gtk_scale_button_get_minus_button (GtkScaleButton *button);
```

Retrieves the minus button of the GtkScaleButton.

Parameters

button	a GtkScaleButton

Returns

the minus button of the GtkScaleButton as a GtkButton.

[transfer none][type Gtk.Button]

Since: 2.14

Types and Values

struct GtkScaleButton

```
struct GtkScaleButton;
```

Property Details

The “adjustment” property

```
  “adjustment”               GtkAdjustment *
```

The GtkAdjustment that contains the current value of this scale button object.

Flags: Read / Write

The “icons” property

```
  “icons”                    GStrv
```

The names of the icons to be used by the scale button. The first item in the array will be used in the button when the current value is the lowest value, the second item for the highest value. All the subsequent icons will be used for all the other values, spread evenly over the range of values.

If there's only one icon name in the `icons` array, it will be used for all the values. If only two icon names are in the `icons` array, the first one will be used for the bottom 50% of the scale, and the second one for the top 50%.

It is recommended to use at least 3 icons so that the GtkScaleButton reflects the current value of the scale better for the users.

Flags: Read / Write

Since: 2.12

The “size” property

```
  “size”                     GtkIconSize
```

The icon size.

Flags: Read / Write

Default value: GTK_ICON_SIZE_SMALL_TOOLBAR

The “value” property

```
  “value”                    gdouble
```

The value of the scale.

Flags: Read / Write

Default value: 0

Signal Details

The “popdown” signal

```
void
user_function (GtkScaleButton *button,
               gpointer        user_data)
```

The ::popdown signal is a keybinding signal which gets emitted to popdown the scale widget.

The default binding for this signal is Escape.

Parameters

button	the object which received the signal
user_data	user data set when the signal handler was connected.

Flags: Action

Since: 2.12

The “popup” signal

```
void
user_function (GtkScaleButton *button,
               gpointer        user_data)
```

The ::popup signal is a keybinding signal which gets emitted to popup the scale widget.

The default bindings for this signal are Space, Enter and Return.

Parameters

button	the object which received the signal
user_data	user data set when the signal handler was connected.

Flags: Action

Since: 2.12

The “value-changed” signal

```
void
user_function (GtkScaleButton *button,
               gdouble         value,
               gpointer        user_data)
```

The ::value-changed signal is emitted when the value field has changed.

Parameters

button	the object which received the signal
value	the new value
user_data	user data set when the signal handler was connected.

Flags: Run Last

Since: 2.12

GtkVolumeButton

GtkVolumeButton — A button which pops up a volume control

Functions

GtkWidget *	gtk_volume_button_new ()

Properties

gboolean	use-symbolic	Read / Write / Construct

Types and Values

struct	GtkVolumeButton

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkContainer
            └── GtkBin
                └── GtkButton
                    └── GtkScaleButton
                        └── GtkVolumeButton
```

Implemented Interfaces

GtkVolumeButton implements AtkImplementorIface, GtkBuildable, GtkActionable, GtkActivatable and GtkOrientable.

Includes

```
#include <gtk/gtk.h>
```

Description

GtkVolumeButton is a subclass of GtkScaleButton that has been tailored for use as a volume control widget with suitable icons, tooltips and accessible labels.

Functions

gtk_volume_button_new ()

```
GtkWidget *
gtk_volume_button_new (void);
```

Creates a GtkVolumeButton, with a range between 0.0 and 1.0, with a stepping of 0.02. Volume values can be obtained and modified using the functions from GtkScaleButton.

Returns

a new GtkVolumeButton

Since: 2.12

Types and Values

struct GtkVolumeButton

```
struct GtkVolumeButton;
```

Property Details

The “use-symbolic” property

“use-symbolic” gboolean

Whether to use symbolic icons as the icons. Note that if the symbolic icons are not available in your installed theme, then the normal (potentially colorful) icons will be used.

Flags: Read / Write / Construct

Default value: TRUE

Since: 3.0

GtkLockButton

GtkLockButton — A widget to unlock or lock privileged operations

Functions

GtkWidget *	gtk_lock_button_new ()
GPermission *	gtk_lock_button_get_permission ()
void	gtk_lock_button_set_permission ()

Properties

GPermission *	permission	Read / Write
gchar *	text-lock	Read / Write / Construct
gchar *	text-unlock	Read / Write / Construct
gchar *	tooltip-lock	Read / Write / Construct
gchar *	tooltip-not-authorized	Read / Write / Construct
gchar *	tooltip-unlock	Read / Write / Construct

Types and Values

struct	GtkLockButton
struct	GtkLockButtonClass

Object Hierarchy

```
GObject
└── GInitiallyUnowned
```

```
        └── GtkWidget
            └── GtkContainer
                └── GtkBin
                    └── GtkButton
                        └── GtkLockButton
```

Implemented Interfaces

GtkLockButton implements AtkImplementorIface, GtkBuildable, GtkActionable and GtkActivatable.

Includes

```
#include <gtk/gtk.h>
```

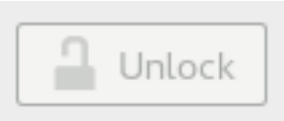
Description

GtkLockButton is a widget that can be used in control panels or preference dialogs to allow users to obtain and revoke authorizations needed to operate the controls. The required authorization is represented by a GPermission object. Concrete implementations of GPermission may use PolicyKit or some other authorization framework. To obtain a PolicyKit-based GPermission, use `polkit_permission_new()`.

If the user is not currently allowed to perform the action, but can obtain the permission, the widget looks like this:

and the user can click the button to request the permission. Depending on the platform, this may pop up an authentication dialog or ask the user to authenticate in some other way. Once the user has obtained the permission, the widget changes to this:

and the permission can be dropped again by clicking the button. If the user is not able to obtain the permission at all, the widget looks like this:



If the user has the permission and cannot drop it, the button is hidden.

The text (and tooltips) that are shown in the various cases can be adjusted with the “text-lock”, “text-unlock”, “tooltip-lock”, “tooltip-unlock” and “tooltip-not-authorized” properties.

Functions

gtk_lock_button_new ()

```
GtkWidget *
gtk_lock_button_new (GPermission *permission);
```

Creates a new lock button which reflects the `permission` .

Parameters

permission	a GPermission.	[allow-none]

Returns

a new GtkLockButton

Since: 3.2

---

gtk_lock_button_get_permission ()

```
GPermission *
gtk_lock_button_get_permission (GtkLockButton *button);
```

Obtains the GPermission object that controls `button` .

Parameters

button	a GtkLockButton

Returns

the GPermission of `button` .

[transfer none]

Since: 3.2

---

gtk_lock_button_set_permission ()

```
void
gtk_lock_button_set_permission (GtkLockButton *button,
                                GPermission *permission);
```

Sets the GPermission object that controls `button` .

Parameters

button	a GtkLockButton	
permission	a GPermission object, or `NULL`.	[allow-none]

Since: 3.2

Types and Values

struct GtkLockButton

```
struct GtkLockButton;
```

---

struct GtkLockButtonClass

```
struct GtkLockButtonClass {
  GtkButtonClass parent_class;
};
```

Members

Property Details

The “permission” property

```
  “permission”               GPermission *
```

The GPermission object controlling this button.

Flags: Read / Write

The “text-lock” property

```
  “text-lock”                gchar *
```

The text to display when prompting the user to lock.

Flags: Read / Write / Construct

Default value: "Lock"

The “text-unlock” property

```
  “text-unlock”              gchar *
```

The text to display when prompting the user to unlock.

Flags: Read / Write / Construct

Default value: "Unlock"

The “tooltip-lock” property

```
  “tooltip-lock”             gchar *
```

The tooltip to display when prompting the user to lock.

Flags: Read / Write / Construct

Default value: "Dialog is unlocked.\nClick to prevent further changes"

The “tooltip-not-authorized” property

```
  “tooltip-not-authorized”   gchar *
```

The tooltip to display when prompting the user cannot obtain authorization.

Flags: Read / Write / Construct

Default value: "System policy prevents changes.\nContact your system administrator"

The “tooltip-unlock” property

```
  “tooltip-unlock”           gchar *
```

The tooltip to display when prompting the user to unlock.

Flags: Read / Write / Construct

Default value: "Dialog is locked.\nClick to make changes"

GtkModelButton

GtkModelButton — A button that uses a GAction as model

Functions

GtkWidget *	gtk_model_button_new ()

Properties

gboolean	active	Read / Write
gboolean	centered	Read / Write
GIcon *	icon	Read / Write
gboolean	iconic	Read / Write
gboolean	inverted	Read / Write
gchar *	menu-name	Read / Write
GtkButtonRole	role	Read / Write
gchar *	text	Read / Write

gboolean	use-markup	Read / Write

Types and Values

	GtkModelButton
enum	GtkButtonRole

Object Hierarchy

```
    GObject
    └── GInitiallyUnowned
        └── GtkWidget
            └── GtkContainer
                └── GtkBin
                    └── GtkButton
                        └── GtkModelButton
```

Implemented Interfaces

GtkModelButton implements AtkImplementorIface, GtkBuildable, GtkActionable and GtkActivatable.

Includes

```
#include <gtk/gtk.h>
```

Description

GtkModelButton is a button class that can use a GAction as its model. In contrast to GtkToggleButton or GtkRadioButton, which can also be backed by a GAction via the “action-name” property, GtkModelButton will adapt its appearance according to the kind of action it is backed by, and appear either as a plain, check or radio button.

Model buttons are used when popovers from a menu model with `gtk_popover_new_from_model()`; they can also be used manually in a GtkPopoverMenu.

When the action is specified via the “action-name” and “action-target” properties, the role of the button (i.e. whether it is a plain, check or radio button) is determined by the type of the action and doesn't have to be explicitly specified with the “role” property.

The content of the button is specified by the “text” and “icon” properties.

The appearance of model buttons can be influenced with the “centered” and “iconic” properties.

Model buttons have built-in support for submenus in GtkPopoverMenu. To make a GtkModelButton that opens a submenu when activated, set the “menu-name” property. To make a button that goes back to the parent menu, you should set the “inverted” property to place the submenu indicator at the opposite side.

Example

```
1   <object class="GtkPopoverMenu">
2     <child>
```

```
3       <object class="GtkBox">
4         <property name="visible">True</property>
5         <property name="margin">10</property>
6         <child>
7           <object class="GtkModelButton">
8             <property name="visible">True</property>
9             <property name="action-name">view.cut</property>
10            <property name="text" translatable="yes">Cut</property>
11          </object>
12        </child>
13        <child>
14          <object class="GtkModelButton">
15            <property name="visible">True</property>
16            <property name="action-name">view.copy</property>
17            <property name="text" translatable="yes">Copy</property>
18          </object>
19        </child>
20        <child>
21          <object class="GtkModelButton">
22            <property name="visible">True</property>
23            <property name="action-name">view.paste</property>
24            <property name="text" translatable="yes">Paste</property>
25          </object>
26        </child>
27      </object>
28    </child>
29  </object>
```

CSS nodes

```
1   modelbutton
2   ├── <child>
3   └── check
1   modelbutton
2   ├── <child>
3   └── radio
1   modelbutton
2   ├── <child>
3   └── arrow
```

GtkModelButton has a main CSS node with name modelbutton, and a subnode, which will have the name check, radio or arrow, depending on the role of the button and whether it has a menu name set.

The subnode is positioned before or after the content nodes and gets the .left or .right style class, depending on where it is located.

```
1   button.model
2   ├── <child>
3   └── check
```

Iconic model buttons (see “iconic”) change the name of their main node to button and add a .model style class to it. The indicator subnode is invisible in this case.

Functions

gtk_model_button_new ()

```
GtkWidget *
```

```
gtk_model_button_new (void);
```
Creates a new GtkModelButton.

Returns

the newly created GtkModelButton widget

Since: 3.16

Types and Values

GtkModelButton

```
typedef struct _GtkModelButton GtkModelButton;
```

---

enum GtkButtonRole

The role specifies the desired appearance of a GtkModelButton.

Members

GTK_BUTTON_ROLE_NORMAL	A plain button
GTK_BUTTON_ROLE_CHECK	A check button
GTK_BUTTON_ROLE_RADIO	A radio button

Property Details

The `"active"` property

```
  "active"                   gboolean
```
The state of the button. This is reflecting the state of the associated GAction.

Flags: Read / Write

Default value: FALSE

Since: 3.16

The `"centered"` property

```
  "centered"                 gboolean
```
Whether to render the button contents centered instead of left-aligned. This property should be set for title-like items.

Flags: Read / Write

Default value: FALSE

Since: 3.16

The “icon” property

“icon” GIcon *

A GIcon that will be used if iconic appearance for the button is desired.

Flags: Read / Write

Since: 3.16

The “iconic” property

“iconic” gboolean

If this property is set, the button will show an icon if one is set. If no icon is set, the text will be used. This is typically used for horizontal sections of linked buttons.

Flags: Read / Write

Default value: FALSE

Since: 3.16

The “inverted” property

“inverted” gboolean

Whether to show the submenu indicator at the opposite side than normal. This property should be set for model buttons that 'go back' to a parent menu.

Flags: Read / Write

Default value: FALSE

Since: 3.16

The “menu-name” property

“menu-name” gchar *

The name of a submenu to open when the button is activated. If this is set, the button should not have an action associated with it.

Flags: Read / Write

Default value: NULL

Since: 3.16

The “role” property

“role” GtkButtonRole

Specifies whether the button is a plain, check or radio button. When “action-name” is set, the role will be determined from the action and does not have to be set explicitly.

Flags: Read / Write

Default value: GTK_BUTTON_ROLE_NORMAL

Since: 3.16

The “text” property

“text” gchar *

The label for the button.

Flags: Read / Write

Default value: ""

Since: 3.16

The “use-markup” property

“use-markup” gboolean

If TRUE, XML tags in the text of the button are interpreted as by pango_parse_markup() to format the enclosed spans of text. If FALSE, the text will be displayed verbatim.

Flags: Read / Write

Default value: FALSE

Since: 3.24

Numeric and Text Data Entry

GtkEntry — A single line text entry field

GtkEntryBuffer — Text buffer for GtkEntry

GtkEntryCompletion — Completion functionality for GtkEntry

GtkScale — A slider widget for selecting a value from a range

GtkSpinButton — Retrieve an integer or floating-point number from the user

GtkSearchEntry — An entry which shows a search icon

GtkSearchBar — A toolbar to integrate a search entry with

GtkEditable — Interface for text-editing widgets

GtkEntry

GtkEntry — A single line text entry field

Functions

GtkWidget *	gtk_entry_new ()
GtkWidget *	gtk_entry_new_with_buffer ()
GtkEntryBuffer *	gtk_entry_get_buffer ()
void	gtk_entry_set_buffer ()
void	gtk_entry_set_text ()
const gchar *	gtk_entry_get_text ()
guint16	gtk_entry_get_text_length ()
void	gtk_entry_get_text_area ()
void	gtk_entry_set_visibility ()
void	gtk_entry_set_invisible_char ()
void	gtk_entry_unset_invisible_char ()
void	gtk_entry_set_max_length ()
gboolean	gtk_entry_get_activates_default ()
gboolean	gtk_entry_get_has_frame ()
const GtkBorder *	gtk_entry_get_inner_border ()
gint	gtk_entry_get_width_chars ()
gint	gtk_entry_get_max_width_chars ()
void	gtk_entry_set_activates_default ()
void	gtk_entry_set_has_frame ()
void	gtk_entry_set_inner_border ()
void	gtk_entry_set_width_chars ()

void	gtk_entry_set_max_width_chars ()
gunichar	gtk_entry_get_invisible_char ()
void	gtk_entry_set_alignment ()
gfloat	gtk_entry_get_alignment ()
void	gtk_entry_set_placeholder_text ()
const gchar *	gtk_entry_get_placeholder_text ()
void	gtk_entry_set_overwrite_mode ()
gboolean	gtk_entry_get_overwrite_mode ()
PangoLayout *	gtk_entry_get_layout ()
void	gtk_entry_get_layout_offsets ()
gint	gtk_entry_layout_index_to_text_index ()
gint	gtk_entry_text_index_to_layout_index ()
void	gtk_entry_set_attributes ()
PangoAttrList *	gtk_entry_get_attributes ()
gint	gtk_entry_get_max_length ()
gboolean	gtk_entry_get_visibility ()
void	gtk_entry_set_completion ()
GtkEntryCompletion *	gtk_entry_get_completion ()
void	gtk_entry_set_cursor_hadjustment ()
GtkAdjustment *	gtk_entry_get_cursor_hadjustment ()
void	gtk_entry_set_progress_fraction ()
gdouble	gtk_entry_get_progress_fraction ()
void	gtk_entry_set_progress_pulse_step ()
gdouble	gtk_entry_get_progress_pulse_step ()
void	gtk_entry_progress_pulse ()
gboolean	gtk_entry_im_context_filter_keypress ()
void	gtk_entry_reset_im_context ()
PangoTabArray *	gtk_entry_get_tabs ()
void	gtk_entry_set_tabs ()
void	gtk_entry_set_icon_from_pixbuf ()
void	gtk_entry_set_icon_from_stock ()
void	gtk_entry_set_icon_from_icon_name ()
void	gtk_entry_set_icon_from_gicon ()
GtkImageType	gtk_entry_get_icon_storage_type ()
GdkPixbuf *	gtk_entry_get_icon_pixbuf ()
const gchar *	gtk_entry_get_icon_stock ()
const gchar *	gtk_entry_get_icon_name ()
GIcon *	gtk_entry_get_icon_gicon ()
void	gtk_entry_set_icon_activatable ()
gboolean	gtk_entry_get_icon_activatable ()
void	gtk_entry_set_icon_sensitive ()
gboolean	gtk_entry_get_icon_sensitive ()
gint	gtk_entry_get_icon_at_pos ()
void	gtk_entry_set_icon_tooltip_text ()
gchar *	gtk_entry_get_icon_tooltip_text ()
void	gtk_entry_set_icon_tooltip_markup ()
gchar *	gtk_entry_get_icon_tooltip_markup ()
void	gtk_entry_set_icon_drag_source ()
gint	gtk_entry_get_current_icon_drag_source ()
void	gtk_entry_get_icon_area ()
void	gtk_entry_set_input_purpose ()
GtkInputPurpose	gtk_entry_get_input_purpose ()

void	gtk_entry_set_input_hints ()
GtkInputHints	gtk_entry_get_input_hints ()
void	gtk_entry_grab_focus_without_selecting ()

Properties

gboolean	activates-default	Read / Write
PangoAttrList *	attributes	Read / Write
GtkEntryBuffer *	buffer	Read / Write / Construct
gboolean	caps-lock-warning	Read / Write
GtkEntryCompletion *	completion	Read / Write
gint	cursor-position	Read
gboolean	editable	Read / Write
gboolean	enable-emoji-completion	Read / Write
gboolean	has-frame	Read / Write
gchar *	im-module	Read / Write
GtkBorder *	inner-border	Read / Write
GtkInputHints	input-hints	Read / Write
GtkInputPurpose	input-purpose	Read / Write
guint	invisible-char	Read / Write
gboolean	invisible-char-set	Read / Write
gint	max-length	Read / Write
gint	max-width-chars	Read / Write
gboolean	overwrite-mode	Read / Write
gchar *	placeholder-text	Read / Write
gboolean	populate-all	Read / Write
gboolean	primary-icon-activatable	Read / Write
GIcon *	primary-icon-gicon	Read / Write
gchar *	primary-icon-name	Read / Write
GdkPixbuf *	primary-icon-pixbuf	Read / Write
gboolean	primary-icon-sensitive	Read / Write
gchar *	primary-icon-stock	Read / Write
GtkImageType	primary-icon-storage-type	Read
gchar *	primary-icon-tooltip-markup	Read / Write
gchar *	primary-icon-tooltip-text	Read / Write
gdouble	progress-fraction	Read / Write
gdouble	progress-pulse-step	Read / Write
gint	scroll-offset	Read
gboolean	secondary-icon-activatable	Read / Write
GIcon *	secondary-icon-gicon	Read / Write
gchar *	secondary-icon-name	Read / Write
GdkPixbuf *	secondary-icon-pixbuf	Read / Write
gboolean	secondary-icon-sensitive	Read / Write
gchar *	secondary-icon-stock	Read / Write
GtkImageType	secondary-icon-storage-type	Read
gchar *	secondary-icon-tooltip-markup	Read / Write
gchar *	secondary-icon-tooltip-text	Read / Write
gint	selection-bound	Read
GtkShadowType	shadow-type	Read / Write
gboolean	show-emoji-icon	Read / Write
PangoTabArray *	tabs	Read / Write

gchar *	text	Read / Write
guint	text-length	Read
gboolean	truncate-multiline	Read / Write
gboolean	visibility	Read / Write
gint	width-chars	Read / Write
gfloat	xalign	Read / Write

Style Properties

gboolean	icon-prelight	Read
GtkBorder *	inner-border	Read
guint	invisible-char	Read
GtkBorder *	progress-border	Read

Signals

void	activate	Action
void	backspace	Action
void	copy-clipboard	Action
void	cut-clipboard	Action
void	delete-from-cursor	Action
void	icon-press	Run Last
void	icon-release	Run Last
void	insert-at-cursor	Action
void	insert-emoji	Action
void	move-cursor	Action
void	paste-clipboard	Action
void	populate-popup	Run Last
void	preedit-changed	Action
void	toggle-overwrite	Action

Types and Values

struct	GtkEntry
struct	GtkEntryClass
enum	GtkEntryIconPosition
enum	GtkInputPurpose
enum	GtkInputHints

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkEntry
            ├── GtkSearchEntry
            └── GtkSpinButton
```

Implemented Interfaces

GtkEntry implements AtkImplementorIface, GtkBuildable, GtkEditable and GtkCellEditable.

Includes

```
#include <gtk/gtk.h>
```

Description

The GtkEntry widget is a single line text entry widget. A fairly large set of key bindings are supported by default. If the entered text is longer than the allocation of the widget, the widget will scroll so that the cursor position is visible.

When using an entry for passwords and other sensitive information, it can be put into “password mode” using `gtk_entry_set_visibility()`. In this mode, entered text is displayed using a “invisible” character. By default, GTK+ picks the best invisible character that is available in the current font, but it can be changed with `gtk_entry_set_invisible_char()`. Since 2.16, GTK+ displays a warning when Caps Lock or input methods might interfere with entering text in a password entry. The warning can be turned off with the “caps-lock-warning” property.

Since 2.16, GtkEntry has the ability to display progress or activity information behind the text. To make an entry display such information, use `gtk_entry_set_progress_fraction()` or `gtk_entry_set_progress_pulse_step()`.

Additionally, GtkEntry can show icons at either side of the entry. These icons can be activatable by clicking, can be set up as drag source and can have tooltips. To add an icon, use `gtk_entry_set_icon_from_gicon()` or one of the various other functions that set an icon from a stock id, an icon name or a pixbuf. To trigger an action when the user clicks an icon, connect to the “icon-press” signal. To allow DND operations from an icon, use `gtk_entry_set_icon_drag_source()`. To set a tooltip on an icon, use `gtk_entry_set_icon_tooltip_text()` or the corresponding function for markup.

Note that functionality or information that is only available by clicking on an icon in an entry may not be accessible at all to users which are not able to use a mouse or other pointing device. It is therefore recommended that any such functionality should also be available by other means, e.g. via the context menu of the entry.

CSS nodes

```
entry[.read-only][.flat][.warning][.error]
├── image.left
├── image.right
├── undershoot.left
├── undershoot.right
├── [selection]
├── [progress[.pulse]]
└── [window.popup]
```

GtkEntry has a main node with the name entry. Depending on the properties of the entry, the style classes .read-only and .flat may appear. The style classes .warning and .error may also be used with entries.

When the entry shows icons, it adds subnodes with the name image and the style class .left or .right, depending on where the icon appears.

When the entry has a selection, it adds a subnode with the name selection.

When the entry shows progress, it adds a subnode with the name progress. The node has the style class .pulse when the shown progress is pulsing.

The CSS node for a context menu is added as a subnode below entry as well.

The undershoot nodes are used to draw the underflow indication when content is scrolled out of view. These nodes get the .left and .right style classes added depending on where the indication is drawn.

When touch is used and touch selection handles are shown, they are using CSS nodes with name cursor-handle. They get the .top or .bottom style class depending on where they are shown in relation to the selection. If there is just a single handle for the text cursor, it gets the style class .insertion-cursor.

Functions

gtk_entry_new ()

```
GtkWidget *
gtk_entry_new (void);
```

Creates a new entry.

Returns

a new GtkEntry.

---

gtk_entry_new_with_buffer ()

```
GtkWidget *
gtk_entry_new_with_buffer (GtkEntryBuffer *buffer);
```

Creates a new entry with the specified text buffer.

Parameters

buffer	The buffer to use for the new GtkEntry.

Returns

a new GtkEntry

Since: 2.18

---

gtk_entry_get_buffer ()

```
GtkEntryBuffer *
gtk_entry_get_buffer (GtkEntry *entry);
```

Get the GtkEntryBuffer object which holds the text for this widget.

Parameters

entry	a GtkEntry

Returns

A GtkEntryBuffer object.

[transfer none]

Since: 2.18

---

gtk_entry_set_buffer ()

```
void
gtk_entry_set_buffer (GtkEntry *entry,
                      GtkEntryBuffer *buffer);
```

Set the GtkEntryBuffer object which holds the text for this widget.

Parameters

entry	a GtkEntry
buffer	a GtkEntryBuffer

Since: 2.18

---

gtk_entry_set_text ()

```
void
gtk_entry_set_text (GtkEntry *entry,
                    const gchar *text);
```

Sets the text in the widget to the given value, replacing the current contents.

See `gtk_entry_buffer_set_text()`.

Parameters

entry	a GtkEntry
text	the new text

---

gtk_entry_get_text ()

```
const gchar *
gtk_entry_get_text (GtkEntry *entry);
```

Retrieves the contents of the entry widget. See also `gtk_editable_get_chars()`.

This is equivalent to getting `entry` 's GtkEntryBuffer and calling `gtk_entry_buffer_get_text()` on it.

Parameters

entry	a GtkEntry

Returns

a pointer to the contents of the widget as a string. This string points to internally allocated storage in the widget and must not be freed, modified or stored.

---

gtk_entry_get_text_length ()

```
guint16
gtk_entry_get_text_length (GtkEntry *entry);
```

Retrieves the current length of the text in `entry` .

This is equivalent to getting `entry` 's GtkEntryBuffer and calling `gtk_entry_buffer_get_length()` on it.

Parameters

entry	a GtkEntry

Returns

the current number of characters in GtkEntry, or 0 if there are none.

Since: 2.14

---

gtk_entry_get_text_area ()

```
void
gtk_entry_get_text_area (GtkEntry *entry,
                         GdkRectangle *text_area);
```

Gets the area where the entry’s text is drawn. This function is useful when drawing something to the entry in a draw callback.

If the entry is not realized, `text_area` is filled with zeros.

See also `gtk_entry_get_icon_area()`.

Parameters

entry	a GtkEntry	
text_area	Return location for the text area.	[out]

Since: 3.0

---

gtk_entry_set_visibility ()

```
void
gtk_entry_set_visibility (GtkEntry *entry,
                          gboolean visible);
```

Sets whether the contents of the entry are visible or not. When visibility is set to FALSE, characters are displayed as the invisible char, and will also appear that way when the text in the entry widget is copied elsewhere.

By default, GTK+ picks the best invisible character available in the current font, but it can be changed with gtk_entry_set_invisible_char().

Note that you probably want to set "input-purpose" to GTK_INPUT_PURPOSE_PASSWORD or GTK_INPUT_PURPOSE_PIN to inform input methods about the purpose of this entry, in addition to setting visibility to FALSE.

Parameters

entry	a GtkEntry
visible	TRUE if the contents of the entry are displayed as plaintext

---

gtk_entry_set_invisible_char ()

```
void
gtk_entry_set_invisible_char (GtkEntry *entry,
                              gunichar ch);
```

Sets the character to use in place of the actual text when gtk_entry_set_visibility() has been called to set text visibility to FALSE. i.e. this is the character used in "password mode" to show the user how many characters have been typed. By default, GTK+ picks the best invisible char available in the current font. If you set the invisible char to 0, then the user will get no feedback at all; there will be no text on the screen as they type.

Parameters

entry	a GtkEntry
ch	a Unicode character

---

gtk_entry_unset_invisible_char ()

```
void
gtk_entry_unset_invisible_char (GtkEntry *entry);
```

Unsets the invisible char previously set with `gtk_entry_set_invisible_char()`. So that the default invisible char is used again.

Parameters

entry	a GtkEntry

Since: 2.16

---

gtk_entry_set_max_length ()

```
void
gtk_entry_set_max_length (GtkEntry *entry,
                          gint max);
```

Sets the maximum allowed length of the contents of the widget. If the current contents are longer than the given length, then they will be truncated to fit.

This is equivalent to getting `entry` 's GtkEntryBuffer and calling `gtk_entry_buffer_set_max_length()` on it.]|

Parameters

entry	a GtkEntry
max	the maximum length of the entry, or 0 for no maximum. (other than the maximum length of entries.) The value passed in will be clamped to the range 0-65536.

---

gtk_entry_get_activates_default ()

```
gboolean
gtk_entry_get_activates_default (GtkEntry *entry);
```

Retrieves the value set by `gtk_entry_set_activates_default()`.

Parameters

entry	a GtkEntry

Returns

`TRUE` if the entry will activate the default widget

---

gtk_entry_get_has_frame ()

```
gboolean
gtk_entry_get_has_frame (GtkEntry *entry);
```

Gets the value set by `gtk_entry_set_has_frame()`.

Parameters

entry	a GtkEntry

Returns

whether the entry has a beveled frame

---

gtk_entry_get_inner_border ()

```
const GtkBorder *
gtk_entry_get_inner_border (GtkEntry *entry);
```

`gtk_entry_get_inner_border` has been deprecated since version 3.4 and should not be used in newly-written code.

Use the standard border and padding CSS properties (through objects like GtkStyleContext and GtkCssProvider); the value returned by this function is ignored by GtkEntry.

This function returns the entry's "inner-border" property. See `gtk_entry_set_inner_border()` for more information.

Parameters

entry	a GtkEntry

Returns

the entry's GtkBorder, or `NULL` if none was set.

[nullable][transfer none]

Since: 2.10

---

gtk_entry_get_width_chars ()

```
gint
gtk_entry_get_width_chars (GtkEntry *entry);
```

Gets the value set by `gtk_entry_set_width_chars()`.

Parameters

entry	a GtkEntry

Returns

number of chars to request space for, or negative if unset

---

gtk_entry_get_max_width_chars ()

```
gint
gtk_entry_get_max_width_chars (GtkEntry *entry);
```

Retrieves the desired maximum width of `entry` , in characters. See `gtk_entry_set_max_width_chars()`.

Parameters

entry	a GtkEntry

Returns

the maximum width of the entry, in characters

Since: 3.12

---

gtk_entry_set_activates_default ()

```
void
gtk_entry_set_activates_default (GtkEntry *entry,
                                 gboolean setting);
```

If `setting` is `TRUE`, pressing Enter in the `entry` will activate the default widget for the window containing the entry. This usually means that the dialog box containing the entry will be closed, since the default widget is usually one of the dialog buttons.

(For experts: if `setting` is `TRUE`, the entry calls `gtk_window_activate_default()` on the window containing the entry, in the default handler for the "activate" signal.)

Parameters

entry	a GtkEntry
setting	`TRUE` to activate window's default widget on Enter keypress

---

gtk_entry_set_has_frame ()

```
void
```

```
gtk_entry_set_has_frame (GtkEntry *entry,
                         gboolean setting);
```

Sets whether the entry has a beveled frame around it.

Parameters

entry	a GtkEntry
setting	new value

---

gtk_entry_set_inner_border ()

```
void
gtk_entry_set_inner_border (GtkEntry *entry,
                            const GtkBorder *border);
```

`gtk_entry_set_inner_border` has been deprecated since version 3.4 and should not be used in newly-written code.

Use the standard border and padding CSS properties (through objects like GtkStyleContext and GtkCssProvider); the value set with this function is ignored by GtkEntry.

Sets `entry`'s inner-border property to `border` , or clears it if `NULL` is passed. The inner-border is the area around the entry's text, but inside its frame.

If set, this property overrides the inner-border style property. Overriding the style-provided border is useful when you want to do in-place editing of some text in a canvas or list widget, where pixel-exact positioning of the entry is important.

Parameters

entry	a GtkEntry	
border	a GtkBorder, or `NULL`.	[allow-none]

Since: 2.10

---

gtk_entry_set_width_chars ()

```
void
gtk_entry_set_width_chars (GtkEntry *entry,
                           gint n_chars);
```

Changes the size request of the entry to be about the right size for `n_chars` characters. Note that it changes the size request, the size can still be affected by how you pack the widget into containers. If `n_chars` is -1, the size reverts to the default entry size.

Parameters

entry	a GtkEntry
n_chars	width in chars

---

gtk_entry_set_max_width_chars ()

```
void
gtk_entry_set_max_width_chars (GtkEntry *entry,
                               gint n_chars);
```

Sets the desired maximum width in characters of `entry` .

Parameters

entry	a GtkEntry
n_chars	the new desired maximum width, in characters

Since: 3.12

---

gtk_entry_get_invisible_char ()

```
gunichar
gtk_entry_get_invisible_char (GtkEntry *entry);
```

Retrieves the character displayed in place of the real characters for entries with visibility set to false. See `gtk_entry_set_invisible_char()`.

Parameters

entry	a GtkEntry

Returns

the current invisible char, or 0, if the entry does not show invisible text at all.

---

gtk_entry_set_alignment ()

```
void
gtk_entry_set_alignment (GtkEntry *entry,
                         gfloat xalign);
```

Sets the alignment for the contents of the entry. This controls the horizontal positioning of the contents when the displayed text is shorter than the width of the entry.

Parameters

entry	a GtkEntry
xalign	The horizontal alignment, from 0 (left) to 1 (right). Reversed for RTL layouts

Since: 2.4

---

gtk_entry_get_alignment ()

```
gfloat
gtk_entry_get_alignment (GtkEntry *entry);
```

Gets the value set by `gtk_entry_set_alignment()`.

Parameters

entry	a GtkEntry

Returns

the alignment

Since: 2.4

---

gtk_entry_set_placeholder_text ()

```
void
gtk_entry_set_placeholder_text (GtkEntry *entry,
                                const gchar *text);
```

Sets text to be displayed in `entry` when it is empty and unfocused. This can be used to give a visual hint of the expected contents of the GtkEntry.

Note that since the placeholder text gets removed when the entry received focus, using this feature is a bit problematic if the entry is given the initial focus in a window. Sometimes this can be worked around by delaying the initial focus setting until the first key event arrives.

Parameters

entry	a GtkEntry
text	a string to be displayed when `entry` is empty and unfocused, or `NULL`. [nullable]

Since: 3.2

---

gtk_entry_get_placeholder_text ()

```
const gchar *
gtk_entry_get_placeholder_text (GtkEntry *entry);
```

Retrieves the text that will be displayed when `entry` is empty and unfocused

Parameters

entry	a GtkEntry

Returns

a pointer to the placeholder text as a string. This string points to internally allocated storage in the widget and must not be freed, modified or stored.

Since: 3.2

---

gtk_entry_set_overwrite_mode ()

```
void
gtk_entry_set_overwrite_mode (GtkEntry *entry,
                              gboolean overwrite);
```

Sets whether the text is overwritten when typing in the GtkEntry.

Parameters

entry	a GtkEntry
overwrite	new value

Since: 2.14

---

gtk_entry_get_overwrite_mode ()

```
gboolean
gtk_entry_get_overwrite_mode (GtkEntry *entry);
```

Gets the value set by `gtk_entry_set_overwrite_mode()`.

Parameters

entry	a GtkEntry

Returns

whether the text is overwritten when typing.

Since: 2.14

---

gtk_entry_get_layout ()

```
PangoLayout *
gtk_entry_get_layout (GtkEntry *entry);
```

Gets the PangoLayout used to display the entry. The layout is useful to e.g. convert text positions to pixel positions, in combination with `gtk_entry_get_layout_offsets()`. The returned layout is owned by the entry and must not be modified or freed by the caller.

Keep in mind that the layout text may contain a preedit string, so `gtk_entry_layout_index_to_text_index()` and `gtk_entry_text_index_to_layout_index()` are needed

to convert byte indices in the layout to byte indices in the entry contents.

Parameters

entry	a GtkEntry

Returns

the PangoLayout for this entry.

[transfer none]

---

gtk_entry_get_layout_offsets ()

```
void
gtk_entry_get_layout_offsets (GtkEntry *entry,
                              gint *x,
                              gint *y);
```

Obtains the position of the PangoLayout used to render text in the entry, in widget coordinates. Useful if you want to line up the text in an entry with some other text, e.g. when using the entry to implement editable cells in a sheet widget.

Also useful to convert mouse events into coordinates inside the PangoLayout, e.g. to take some action if some part of the entry text is clicked.

Note that as the user scrolls around in the entry the offsets will change; you'll need to connect to the "notify::scroll-offset" signal to track this. Remember when using the PangoLayout functions you need to convert to and from pixels using `PANGO_PIXELS()` or PANGO_SCALE.

Keep in mind that the layout text may contain a preedit string, so `gtk_entry_layout_index_to_text_index()` and `gtk_entry_text_index_to_layout_index()` are needed to convert byte indices in the layout to byte indices in the entry contents.

Parameters

entry	a GtkEntry	
x	location to store X offset of layout, or `NULL`.	[out][allow-none]
y	location to store Y offset of layout, or `NULL`.	[out][allow-none]

---

gtk_entry_layout_index_to_text_index ()

```
gint
gtk_entry_layout_index_to_text_index (GtkEntry *entry,
                                      gint layout_index);
```

Converts from a position in the entry's PangoLayout (returned by `gtk_entry_get_layout()`) to a position in the entry contents (returned by `gtk_entry_get_text()`).

Parameters

entry	a GtkEntry
layout_index	byte index into the entry layout text

Returns

byte index into the entry contents

---

gtk_entry_text_index_to_layout_index ()

```
gint
gtk_entry_text_index_to_layout_index (GtkEntry *entry,
                                      gint text_index);
```

Converts from a position in the entry contents (returned by `gtk_entry_get_text()`) to a position in the entry's PangoLayout (returned by `gtk_entry_get_layout()`, with text retrieved via `pango_layout_get_text()`).

Parameters

entry	a GtkEntry
text_index	byte index into the entry contents

Returns

byte index into the entry layout text

---

gtk_entry_set_attributes ()

```
void
gtk_entry_set_attributes (GtkEntry *entry,
                          PangoAttrList *attrs);
```

Sets a PangoAttrList; the attributes in the list are applied to the entry text.

Parameters

entry	a GtkEntry
attrs	a PangoAttrList

Since: 3.6

---

gtk_entry_get_attributes ()

```
PangoAttrList *
gtk_entry_get_attributes (GtkEntry *entry);
```

Gets the attribute list that was set on the entry using `gtk_entry_set_attributes()`, if any.

Parameters

entry	a GtkEntry

Returns

the attribute list, or `NULL` if none was set.

[transfer none][nullable]

Since: 3.6

---

gtk_entry_get_max_length ()

```
gint
gtk_entry_get_max_length (GtkEntry *entry);
```

Retrieves the maximum allowed length of the text in `entry` . See `gtk_entry_set_max_length()`.

This is equivalent to getting `entry` 's GtkEntryBuffer and calling `gtk_entry_buffer_get_max_length()` on it.

Parameters

entry	a GtkEntry

Returns

the maximum allowed number of characters in GtkEntry, or 0 if there is no maximum.

---

gtk_entry_get_visibility ()

```
gboolean
gtk_entry_get_visibility (GtkEntry *entry);
```

Retrieves whether the text in `entry` is visible. See `gtk_entry_set_visibility()`.

Parameters

entry	a GtkEntry

Returns

`TRUE` if the text is currently visible

---

gtk_entry_set_completion ()

```
void
gtk_entry_set_completion (GtkEntry *entry,
                          GtkEntryCompletion *completion);
```

Sets `completion` to be the auxiliary completion object to use with `entry` . All further configuration of the completion mechanism is done on `completion` using the GtkEntryCompletion API. Completion is disabled if `completion` is set to `NULL`.

Parameters

entry	A GtkEntry	
completion	The GtkEntryCompletion or `NULL`.	[allow-none]

Since: 2.4

---

gtk_entry_get_completion ()

```
GtkEntryCompletion *
gtk_entry_get_completion (GtkEntry *entry);
```

Returns the auxiliary completion object currently in use by `entry` .

Parameters

entry	A GtkEntry

Returns

The auxiliary completion object currently in use by `entry` .

[transfer none]

Since: 2.4

---

gtk_entry_set_cursor_hadjustment ()

```
void
gtk_entry_set_cursor_hadjustment (GtkEntry *entry,
                                  GtkAdjustment *adjustment);
```

Hooks up an adjustment to the cursor position in an entry, so that when the cursor is moved, the adjustment is scrolled to show that position. See `gtk_scrolled_window_get_hadjustment()` for a typical way of obtaining the adjustment.

The adjustment has to be in pixel units and in the same coordinate system as the entry.

Parameters

entry	a GtkEntry	
adjustment	an adjustment which should be adjusted when the cursor is moved, or `NULL`.	[nullable]

Since: 2.12

---

gtk_entry_get_cursor_hadjustment ()

```
GtkAdjustment *
gtk_entry_get_cursor_hadjustment (GtkEntry *entry);
```

Retrieves the horizontal cursor adjustment for the entry. See `gtk_entry_set_cursor_hadjustment()`.

Parameters

entry	a GtkEntry

Returns

the horizontal cursor adjustment, or `NULL` if none has been set.

[transfer none][nullable]

Since: 2.12

---

gtk_entry_set_progress_fraction ()

```
void
gtk_entry_set_progress_fraction (GtkEntry *entry,
                                 gdouble fraction);
```

Causes the entry's progress indicator to "fill in" the given fraction of the bar. The fraction should be between 0.0 and 1.0, inclusive.

Parameters

entry	a GtkEntry	
fraction	fraction of the task that's been completed	

Since: 2.16

---

gtk_entry_get_progress_fraction ()

```
gdouble
gtk_entry_get_progress_fraction (GtkEntry *entry);
```

Returns the current fraction of the task that's been completed. See `gtk_entry_set_progress_fraction()`.

Parameters

entry	a GtkEntry

Returns

a fraction from 0.0 to 1.0

Since: 2.16

---

gtk_entry_set_progress_pulse_step ()

```
void
gtk_entry_set_progress_pulse_step (GtkEntry *entry,
                                   gdouble fraction);
```

Sets the fraction of total entry width to move the progress bouncing block for each call to `gtk_entry_progress_pulse()`.

Parameters

entry	a GtkEntry
fraction	fraction between 0.0 and 1.0

Since: 2.16

---

gtk_entry_get_progress_pulse_step ()

```
gdouble
gtk_entry_get_progress_pulse_step (GtkEntry *entry);
```

Retrieves the pulse step set with `gtk_entry_set_progress_pulse_step()`.

Parameters

entry	a GtkEntry

Returns

a fraction from 0.0 to 1.0

Since: 2.16

---

gtk_entry_progress_pulse ()

```
void
```

```
gtk_entry_progress_pulse (GtkEntry *entry);
```
Indicates that some progress is made, but you don’t know how much. Causes the entry’s progress indicator to enter “activity mode,” where a block bounces back and forth. Each call to `gtk_entry_progress_pulse()` causes the block to move by a little bit (the amount of movement per pulse is determined by `gtk_entry_set_progress_pulse_step()`).

Parameters

entry	a GtkEntry	

Since: 2.16

---

gtk_entry_im_context_filter_keypress ()

```
gboolean
gtk_entry_im_context_filter_keypress (GtkEntry *entry,
                                      GdkEventKey *event);
```
Allow the GtkEntry input method to internally handle key press and release events. If this function returns `TRUE`, then no further processing should be done for this key event. See `gtk_im_context_filter_keypress()`.

Note that you are expected to call this function from your handler when overriding key event handling. This is needed in the case when you need to insert your own key handling between the input method and the default key event handling of the GtkEntry. See `gtk_text_view_reset_im_context()` for an example of use.

Parameters

entry	a GtkEntry	
event	the key event.	[type Gdk.EventKey]

Returns

`TRUE` if the input method handled the key event.

Since: 2.22

---

gtk_entry_reset_im_context ()

```
void
gtk_entry_reset_im_context (GtkEntry *entry);
```
Reset the input method context of the entry if needed.

This can be necessary in the case where modifying the buffer would confuse on-going input method behavior.

Parameters

entry	a GtkEntry	

Since: 2.22

---

gtk_entry_get_tabs ()

```
PangoTabArray *
gtk_entry_get_tabs (GtkEntry *entry);
```

Gets the tabstops that were set on the entry using `gtk_entry_set_tabs()`, if any.

Parameters

entry	a GtkEntry

Returns

the tabstops, or `NULL` if none was set.

[nullable][transfer none]

Since: 3.10

---

gtk_entry_set_tabs ()

```
void
gtk_entry_set_tabs (GtkEntry *entry,
                    PangoTabArray *tabs);
```

Sets a PangoTabArray; the tabstops in the array are applied to the entry text.

Parameters

entry	a GtkEntry
tabs	a PangoTabArray

Since: 3.10

---

gtk_entry_set_icon_from_pixbuf ()

```
void
gtk_entry_set_icon_from_pixbuf (GtkEntry *entry,
                                GtkEntryIconPosition icon_pos,
                                GdkPixbuf *pixbuf);
```

Sets the icon shown in the specified position using a pixbuf.

If `pixbuf` is `NULL`, no icon will be shown in the specified position.

Parameters

entry	a GtkEntry	
icon_pos	Icon position	
pixbuf	A GdkPixbuf, or `NULL`.	[allow-none]

Since: 2.16

---

gtk_entry_set_icon_from_stock ()

```
void
gtk_entry_set_icon_from_stock (GtkEntry *entry,
                               GtkEntryIconPosition icon_pos,
                               const gchar *stock_id);
```

`gtk_entry_set_icon_from_stock` has been deprecated since version 3.10 and should not be used in newly-written code.

Use `gtk_entry_set_icon_from_icon_name()` instead.

Sets the icon shown in the entry at the specified position from a stock image.

If `stock_id` is `NULL`, no icon will be shown in the specified position.

Parameters

entry	A GtkEntry	
icon_pos	Icon position	
stock_id	The name of the stock item, or `NULL`.	[allow-none]

Since: 2.16

---

gtk_entry_set_icon_from_icon_name ()

```
void
gtk_entry_set_icon_from_icon_name (GtkEntry *entry,
                                   GtkEntryIconPosition icon_pos,
                                   const gchar *icon_name);
```

Sets the icon shown in the entry at the specified position from the current icon theme.

If the icon name isn't known, a "broken image" icon will be displayed instead.

If `icon_name` is `NULL`, no icon will be shown in the specified position.

Parameters

entry	A GtkEntry	
icon_pos	The position at which to set the icon	
icon_name	An icon name, or `NULL`.	[allow-none]

Since: 2.16

---

gtk_entry_set_icon_from_gicon ()

```
void
```

```
gtk_entry_set_icon_from_gicon (GtkEntry *entry,
                               GtkEntryIconPosition icon_pos,
                               GIcon *icon);
```

Sets the icon shown in the entry at the specified position from the current icon theme. If the icon isn't known, a "broken image" icon will be displayed instead.

If `icon` is `NULL`, no icon will be shown in the specified position.

Parameters

entry	A GtkEntry	
icon_pos	The position at which to set the icon	
icon	The icon to set, or `NULL`.	[allow-none]

Since: 2.16

---

gtk_entry_get_icon_storage_type ()

```
GtkImageType
gtk_entry_get_icon_storage_type (GtkEntry *entry,
                                 GtkEntryIconPosition icon_pos);
```

Gets the type of representation being used by the icon to store image data. If the icon has no image data, the return value will be `GTK_IMAGE_EMPTY`.

Parameters

entry	a GtkEntry
icon_pos	Icon position

Returns

image representation being used

Since: 2.16

---

gtk_entry_get_icon_pixbuf ()

```
GdkPixbuf *
gtk_entry_get_icon_pixbuf (GtkEntry *entry,
                           GtkEntryIconPosition icon_pos);
```

Retrieves the image used for the icon.

Unlike the other methods of setting and getting icon data, this method will work regardless of whether the icon was set using a GdkPixbuf, a GIcon, a stock item, or an icon name.

Parameters

entry	A GtkEntry

icon_pos	Icon position

Returns

A GdkPixbuf, or `NULL` if no icon is set for this position.

[transfer none][nullable]

Since: 2.16

---

gtk_entry_get_icon_stock ()

```
const gchar *
gtk_entry_get_icon_stock (GtkEntry *entry,
                          GtkEntryIconPosition icon_pos);
```

`gtk_entry_get_icon_stock` has been deprecated since version 3.10 and should not be used in newly-written code.

Use `gtk_entry_get_icon_name()` instead.

Retrieves the stock id used for the icon, or `NULL` if there is no icon or if the icon was set by some other method (e.g., by pixbuf, icon name or gicon).

Parameters

entry	A GtkEntry
icon_pos	Icon position

Returns

A stock id, or `NULL` if no icon is set or if the icon wasn't set from a stock id

Since: 2.16

---

gtk_entry_get_icon_name ()

```
const gchar *
gtk_entry_get_icon_name (GtkEntry *entry,
                         GtkEntryIconPosition icon_pos);
```

Retrieves the icon name used for the icon, or `NULL` if there is no icon or if the icon was set by some other method (e.g., by pixbuf, stock or gicon).

Parameters

entry	A GtkEntry
icon_pos	Icon position

Returns

An icon name, or `NULL` if no icon is set or if the icon wasn't set from an icon name.

[nullable]

Since: 2.16

---

gtk_entry_get_icon_gicon ()

```
GIcon *
gtk_entry_get_icon_gicon (GtkEntry *entry,
                          GtkEntryIconPosition icon_pos);
```

Retrieves the GIcon used for the icon, or `NULL` if there is no icon or if the icon was set by some other method (e.g., by stock, pixbuf, or icon name).

Parameters

entry	A GtkEntry
icon_pos	Icon position

Returns

A GIcon, or `NULL` if no icon is set or if the icon is not a GIcon.

[transfer none][nullable]

Since: 2.16

---

gtk_entry_set_icon_activatable ()

```
void
gtk_entry_set_icon_activatable (GtkEntry *entry,
                                GtkEntryIconPosition icon_pos,
                                gboolean activatable);
```

Sets whether the icon is activatable.

Parameters

entry	A GtkEntry
icon_pos	Icon position
activatable	`TRUE` if the icon should be activatable

Since: 2.16

---

gtk_entry_get_icon_activatable ()

```
gboolean
gtk_entry_get_icon_activatable (GtkEntry *entry,
                                GtkEntryIconPosition icon_pos);
```

Returns whether the icon is activatable.

Parameters

entry	a GtkEntry
icon_pos	Icon position

Returns

`TRUE` if the icon is activatable.

Since: 2.16

---

gtk_entry_set_icon_sensitive ()

```
void
gtk_entry_set_icon_sensitive (GtkEntry *entry,
                              GtkEntryIconPosition icon_pos,
                              gboolean sensitive);
```

Sets the sensitivity for the specified icon.

Parameters

entry	A GtkEntry
icon_pos	Icon position
sensitive	Specifies whether the icon should appear sensitive or insensitive

Since: 2.16

---

gtk_entry_get_icon_sensitive ()

```
gboolean
gtk_entry_get_icon_sensitive (GtkEntry *entry,
                              GtkEntryIconPosition icon_pos);
```

Returns whether the icon appears sensitive or insensitive.

Parameters

entry	a GtkEntry
icon_pos	Icon position

Returns

TRUE if the icon is sensitive.

Since: 2.16

---

gtk_entry_get_icon_at_pos ()

```
gint
gtk_entry_get_icon_at_pos (GtkEntry *entry,
                           gint x,
                           gint y);
```

Finds the icon at the given position and return its index. The position's coordinates are relative to the entry 's top left corner. If x , y doesn't lie inside an icon, -1 is returned. This function is intended for use in a "query-tooltip" signal handler.

Parameters

entry	a GtkEntry
x	the x coordinate of the position to find
y	the y coordinate of the position to find

Returns

the index of the icon at the given position, or -1

Since: 2.16

---

gtk_entry_set_icon_tooltip_text ()

```
void
gtk_entry_set_icon_tooltip_text (GtkEntry *entry,
                                 GtkEntryIconPosition icon_pos,
                                 const gchar *tooltip);
```

Sets tooltip as the contents of the tooltip for the icon at the specified position.

Use NULL for tooltip to remove an existing tooltip.

See also gtk_widget_set_tooltip_text() and gtk_entry_set_icon_tooltip_markup().

If you unset the widget tooltip via gtk_widget_set_tooltip_text() or gtk_widget_set_tooltip_markup(), this sets GtkWidget:has-tooltip to FALSE, which suppresses icon tooltips too. You can resolve this by then calling gtk_widget_set_has_tooltip() to set GtkWidget:has-tooltip back to TRUE, or setting at least one non-empty tooltip on any icon achieves the same result.

Parameters

entry	a GtkEntry	
icon_pos	the icon position	
tooltip	the contents of the tooltip for the icon, or NULL.	[allow-none]

Since: 2.16

---

gtk_entry_get_icon_tooltip_text ()

```
gchar *
gtk_entry_get_icon_tooltip_text (GtkEntry *entry,
                                 GtkEntryIconPosition icon_pos);
```

Gets the contents of the tooltip on the icon at the specified position in `entry` .

Parameters

entry	a GtkEntry
icon_pos	the icon position

Returns

the tooltip text, or `NULL`. Free the returned string with `g_free()` when done.

[nullable]

Since: 2.16

---

gtk_entry_set_icon_tooltip_markup ()

```
void
gtk_entry_set_icon_tooltip_markup (GtkEntry *entry,
                                   GtkEntryIconPosition icon_pos,
                                   const gchar *tooltip);
```

Sets `tooltip` as the contents of the tooltip for the icon at the specified position. `tooltip` is assumed to be marked up with the Pango text markup language.

Use `NULL` for `tooltip` to remove an existing tooltip.

See also `gtk_widget_set_tooltip_markup()` and `gtk_entry_set_icon_tooltip_text()`.

Parameters

entry	a GtkEntry	
icon_pos	the icon position	
tooltip	the contents of the tooltip for the icon, or NULL.	[allow-none]

Since: 2.16

---

gtk_entry_get_icon_tooltip_markup ()

```
gchar *
gtk_entry_get_icon_tooltip_markup (GtkEntry *entry,
                                   GtkEntryIconPosition icon_pos);
```

Gets the contents of the tooltip on the icon at the specified position in `entry`.

Parameters

entry	a GtkEntry
icon_pos	the icon position

Returns

the tooltip text, or `NULL`. Free the returned string with `g_free()` when done.

[nullable]

Since: 2.16

---

gtk_entry_set_icon_drag_source ()

```
void
gtk_entry_set_icon_drag_source (GtkEntry *entry,
                                GtkEntryIconPosition icon_pos,
                                GtkTargetList *target_list,
                                GdkDragAction actions);
```

Sets up the icon at the given position so that GTK+ will start a drag operation when the user clicks and drags the icon.

To handle the drag operation, you need to connect to the usual "drag-data-get" (or possibly "drag-data-delete") signal, and use `gtk_entry_get_current_icon_drag_source()` in your signal handler to find out if the drag was started from an icon.

By default, GTK+ uses the icon as the drag icon. You can use the "drag-begin" signal to set a different icon. Note that you have to use `g_signal_connect_after()` to ensure that your signal handler gets executed after the default handler.

Parameters

entry	a GtkEntry
icon_pos	icon position
target_list	the targets (data formats) in which the data can be provided
actions	a bitmask of the allowed drag actions

Since: 2.16

---

gtk_entry_get_current_icon_drag_source ()

```
gint
gtk_entry_get_current_icon_drag_source
                                (GtkEntry *entry);
```

Returns the index of the icon which is the source of the current DND operation, or -1.

This function is meant to be used in a “drag-data-get” callback.

Parameters

entry	a GtkEntry

Returns

index of the icon which is the source of the current DND operation, or -1.

Since: 2.16

---

gtk_entry_get_icon_area ()

```
void
gtk_entry_get_icon_area (GtkEntry *entry,
                         GtkEntryIconPosition icon_pos,
                         GdkRectangle *icon_area);
```

Gets the area where entry’s icon at `icon_pos` is drawn. This function is useful when drawing something to the entry in a draw callback.

If the entry is not realized or has no icon at the given position, `icon_area` is filled with zeros. Otherwise, `icon_area` will be filled with the icon’s allocation, relative to `entry` ’s allocation.

See also `gtk_entry_get_text_area()`

Parameters

entry	A GtkEntry	
icon_pos	Icon position	
icon_area	Return location for the icon’s area.	[out]

Since: 3.0

---

gtk_entry_set_input_purpose ()

```
void
gtk_entry_set_input_purpose (GtkEntry *entry,
                             GtkInputPurpose purpose);
```

Sets the “input-purpose” property which can be used by on-screen keyboards and other input methods to adjust their behaviour.

Parameters

entry	a GtkEntry
purpose	the purpose

Since: 3.6

---

gtk_entry_get_input_purpose ()

```
GtkInputPurpose
gtk_entry_get_input_purpose (GtkEntry *entry);
```

Gets the value of the “input-purpose” property.

Parameters

entry	a GtkEntry

Since: 3.6

---

gtk_entry_set_input_hints ()

```
void
gtk_entry_set_input_hints (GtkEntry *entry,
                           GtkInputHints hints);
```

Sets the “input-hints” property, which allows input methods to fine-tune their behaviour.

Parameters

entry	a GtkEntry
hints	the hints

Since: 3.6

---

gtk_entry_get_input_hints ()

```
GtkInputHints
gtk_entry_get_input_hints (GtkEntry *entry);
```

Gets the value of the “input-hints” property.

Parameters

entry	a GtkEntry

Since: 3.6

---

gtk_entry_grab_focus_without_selecting ()

```
void
gtk_entry_grab_focus_without_selecting
                                (GtkEntry *entry);
```

Causes `entry` to have keyboard focus.

It behaves like `gtk_widget_grab_focus()`, except that it doesn't select the contents of the entry. You only want to call this on some special entries which the user usually doesn't want to replace all text in, such as search-as-you-type entries.

Parameters

entry	a GtkEntry

Since: 3.16

Types and Values

struct GtkEntry

```
struct GtkEntry;
```

---

struct GtkEntryClass

```
struct GtkEntryClass {
  GtkWidgetClass parent_class;

  /* Hook to customize right-click popup */
  void (* populate_popup)   (GtkEntry       *entry,
                             GtkWidget      *popup);

  /* Action signals
   */
  void (* activate)           (GtkEntry             *entry);
  void (* move_cursor)        (GtkEntry             *entry,
                          GtkMovementStep       step,
                          gint                  count,
                          gboolean              extend_selection);
  void (* insert_at_cursor)   (GtkEntry             *entry,
                          const gchar          *str);
  void (* delete_from_cursor) (GtkEntry             *entry,
                          GtkDeleteType         type,
                          gint                  count);
  void (* backspace)          (GtkEntry             *entry);
  void (* cut_clipboard)      (GtkEntry             *entry);
  void (* copy_clipboard)     (GtkEntry             *entry);
  void (* paste_clipboard)    (GtkEntry             *entry);
  void (* toggle_overwrite)   (GtkEntry             *entry);

  /* hooks to add other objects beside the entry (like in GtkSpinButton) */
  void (* get_text_area_size) (GtkEntry       *entry,
                          gint           *x,
                          gint           *y,
```

```
                        gint           *width,
                        gint           *height);
  void (* get_frame_size)     (GtkEntry       *entry,
                               gint           *x,
                               gint           *y,
                        gint           *width,
                        gint           *height);
  void (* insert_emoji)       (GtkEntry             *entry);
};
```

Class structure for GtkEntry. All virtual functions have a default implementation. Derived classes may set the virtual function pointers for the signal handlers to `NULL`, but must keep `get_text_area_size` and `get_frame_size` non-`NULL`; either use the default implementation, or provide a custom one.

Members

`populate_popup` ()	Class handler for the "populate-popup" signal. If non-`NULL`, this will be called to add additional entries to the context menu when it is displayed.
`activate` ()	Class handler for the "activate" signal. The default implementation calls `gtk_window_activate_default()` on the entry's top-level window.
`move_cursor` ()	Class handler for the "move-cursor" signal. The default implementation specifies the standard GtkEntry cursor movement behavior.
`insert_at_cursor` ()	Class handler for the "insert-at-cursor" signal. The default implementation inserts text at the cursor.
`delete_from_cursor` ()	Class handler for the "delete-from-cursor" signal. The default implementation deletes the selection or the specified number of characters or words.
`backspace` ()	Class handler for the "backspace" signal. The default implementation deletes the selection or a single character or word.
`cut_clipboard` ()	Class handler for the "cut-clipboard" signal. The default implementation cuts the selection, if one exists.
`copy_clipboard` ()	Class handler for the "copy-clipboard" signal. The default implementation copies the selection, if one exists.
`paste_clipboard` ()	Class handler for the "paste-clipboard" signal. The default

	implementation pastes at the current cursor position or over the current selection if one exists.
`toggle_overwrite` ()	Class handler for the “toggle-overwrite” signal. The default implementation toggles overwrite mode and blinks the cursor.
`get_text_area_size` ()	Calculate the size of the text area, which is its allocated width and requested height, minus space for margins and borders. This virtual function must be non-`NULL`.
`get_frame_size` ()	Calculate the size of the text area frame, which is its allocated width and requested height, minus space for margins and borders, and taking baseline and text height into account. This virtual function must be non-`NULL`.
`insert_emoji` ()	

---

enum GtkEntryIconPosition

Specifies the side of the entry at which an icon is placed.

Members

GTK_ENTRY_ICON_PRIMARY	At the beginning of the entry (depending on the text direction).
GTK_ENTRY_ICON_SECONDARY	At the end of the entry (depending on the text direction).

Since: 2.16

---

enum GtkInputPurpose

Describes primary purpose of the input widget. This information is useful for on-screen keyboards and similar input methods to decide which keys should be presented to the user.

Note that the purpose is not meant to impose a totally strict rule about allowed characters, and does not replace input validation. It is fine for an on-screen keyboard to let the user override the character set restriction that is expressed by the purpose. The application is expected to validate the entry contents, even if it specified a purpose.

The difference between `GTK_INPUT_PURPOSE_DIGITS` and `GTK_INPUT_PURPOSE_NUMBER` is that the former accepts only digits while the latter also some punctuation (like commas or points, plus, minus) and “e” or “E” as in 3.14E+000.

This enumeration may be extended in the future; input methods should interpret unknown values as “free form”.

Members

GTK_INPUT_PURPOSE_FREE_FORM	Allow any character
GTK_INPUT_PURPOSE_ALPHA	Allow only alphabetic characters
GTK_INPUT_PURPOSE_DIGITS	Allow only digits
GTK_INPUT_PURPOSE_NUMBER	Edited field expects numbers
GTK_INPUT_PURPOSE_PHONE	Edited field expects phone number
GTK_INPUT_PURPOSE_URL	Edited field expects URL
GTK_INPUT_PURPOSE_EMAIL	Edited field expects email address
GTK_INPUT_PURPOSE_NAME	Edited field expects the name of a person
GTK_INPUT_PURPOSE_PASSWORD	Like `GTK_INPUT_PURPOSE_FREE_FORM`, but characters are hidden
GTK_INPUT_PURPOSE_PIN	Like `GTK_INPUT_PURPOSE_DIGITS`, but characters are hidden

Since: 3.6

---

enum GtkInputHints

Describes hints that might be taken into account by input methods or applications. Note that input methods may already tailor their behaviour according to the GtkInputPurpose of the entry.

Some common sense is expected when using these flags - mixing `GTK_INPUT_HINT_LOWERCASE` with any of the uppercase hints makes no sense.

This enumeration may be extended in the future; input methods should ignore unknown values.

Members

GTK_INPUT_HINT_NONE	No special behaviour suggested
GTK_INPUT_HINT_SPELLCHECK	Suggest checking for typos
GTK_INPUT_HINT_NO_SPELLCHECK	Suggest not checking for typos
GTK_INPUT_HINT_WORD_COMPLETION	Suggest word completion
GTK_INPUT_HINT_LOWERCASE	Suggest to convert all text to lowercase
GTK_INPUT_HINT_UPPERCASE_CHARS	Suggest to capitalize all text
GTK_INPUT_HINT_UPPERCASE_WORDS	Suggest to capitalize the first character of each word
GTK_INPUT_HINT_UPPERCASE_SENTENCES	Suggest to capitalize the first word of each sentence
GTK_INPUT_HINT_INHIBIT_OSK	Suggest to not show an onscreen keyboard (e.g for a calculator that already has all the keys).

GTK_INPUT_HINT_VERTICAL_WRITING	The text is vertical. Since 3.18
GTK_INPUT_HINT_EMOJI	Suggest offering Emoji support. Since 3.22.20
GTK_INPUT_HINT_NO_EMOJI	Suggest not offering Emoji support. Since 3.22.20

Since: 3.6

Property Details

The `"activates-default"` property

`"activates-default"` `gboolean`

Whether to activate the default widget (such as the default button in a dialog) when Enter is pressed.

Flags: Read / Write

Default value: FALSE

The `"attributes"` property

`"attributes"` `PangoAttrList *`

A list of Pango attributes to apply to the text of the entry.

This is mainly useful to change the size or weight of the text.

The PangoAttribute's `start_index` and `end_index` must refer to the GtkEntryBuffer text, i.e. without the preedit string.

Flags: Read / Write

Since: 3.6

The `"buffer"` property

`"buffer"` `GtkEntryBuffer *`

Text buffer object which actually stores entry text.

Flags: Read / Write / Construct

The `"caps-lock-warning"` property

`"caps-lock-warning"` `gboolean`

Whether password entries will show a warning when Caps Lock is on.

Note that the warning is shown using a secondary icon, and thus does not work if you are using the secondary icon position for some other purpose.

Flags: Read / Write

Default value: TRUE

Since: 2.16

The “completion” property

“completion” GtkEntryCompletion *

The auxiliary completion object to use with the entry.

Flags: Read / Write

Since: 3.2

The “cursor-position” property

“cursor-position” gint

The current position of the insertion cursor in chars.

Flags: Read

Allowed values: [0,65535]

Default value: 0

The “editable” property

“editable” gboolean

Whether the entry contents can be edited.

Flags: Read / Write

Default value: TRUE

The “enable-emoji-completion” property

“enable-emoji-completion” gboolean

Whether to suggest Emoji replacements.

Flags: Read / Write

Default value: FALSE

The “has-frame” property

“has-frame” gboolean

FALSE removes outside bevel from entry.

Flags: Read / Write

Default value: TRUE

The “im-module” property

“im-module” gchar *

Which IM (input method) module should be used for this entry. See GtkIMContext.

Setting this to a non-NULL value overrides the system-wide IM module setting. See the GtkSettings “gtk-im-module” property.

Flags: Read / Write

Default value: NULL

Since: 2.16

The “inner-border” property

“inner-border” GtkBorder *

Sets the text area's border between the text and the frame.

GtkEntry:inner-border has been deprecated since version 3.4 and should not be used in newly-written code.

Use the standard border and padding CSS properties (through objects like GtkStyleContext and GtkCssProvider); the value of this style property is ignored.

Flags: Read / Write

Since: 2.10

The “input-hints” property

“input-hints” GtkInputHints

Additional hints (beyond “input-purpose”) that allow input methods to fine-tune their behaviour.

Flags: Read / Write

Since: 3.6

The “input-purpose” property

“input-purpose” GtkInputPurpose

The purpose of this text field.

This property can be used by on-screen keyboards and other input methods to adjust their behaviour.

Note that setting the purpose to `GTK_INPUT_PURPOSE_PASSWORD` or `GTK_INPUT_PURPOSE_PIN` is independent from setting “visibility”.

Flags: Read / Write

Default value: GTK_INPUT_PURPOSE_FREE_FORM

Since: 3.6

The `“invisible-char”` property

`“invisible-char”` `guint`

The invisible character is used when masking entry contents (in \"password mode\")"). When it is not explicitly set with the “invisible-char” property, GTK+ determines the character to use from a list of possible candidates, depending on availability in the current font.

This style property allows the theme to prepend a character to the list of candidates.

Flags: Read / Write

Default value: '*'

Since: 2.18

The `“invisible-char-set”` property

`“invisible-char-set”` `gboolean`

Whether the invisible char has been set for the GtkEntry.

Flags: Read / Write

Default value: FALSE

Since: 2.16

The `“max-length”` property

`“max-length”` `gint`

Maximum number of characters for this entry. Zero if no maximum.

Flags: Read / Write

Allowed values: [0,65535]

Default value: 0

The “max-width-chars” property

“max-width-chars” gint

The desired maximum width of the entry, in characters. If this property is set to -1, the width will be calculated automatically.

Flags: Read / Write

Allowed values: >= -1

Default value: -1

Since: 3.12

The “overwrite-mode” property

“overwrite-mode” gboolean

If text is overwritten when typing in the GtkEntry.

Flags: Read / Write

Default value: FALSE

Since: 2.14

The “placeholder-text” property

“placeholder-text” gchar *

The text that will be displayed in the GtkEntry when it is empty and unfocused.

Flags: Read / Write

Default value: NULL

Since: 3.2

The “populate-all” property

“populate-all” gboolean

If :populate-all is TRUE, the “populate-popup” signal is also emitted for touch popups.

Flags: Read / Write

Default value: FALSE

Since: 3.8

The “primary-icon-activatable” property

“primary-icon-activatable” gboolean

Whether the primary icon is activatable.

GTK+ emits the “icon-press” and “icon-release” signals only on sensitive, activatable icons.

Sensitive, but non-activatable icons can be used for purely informational purposes.

Flags: Read / Write

Default value: TRUE

Since: 2.16

The “primary-icon-gicon” property

“primary-icon-gicon” GIcon *

The GIcon to use for the primary icon for the entry.

Flags: Read / Write

Since: 2.16

The “primary-icon-name” property

“primary-icon-name” gchar *

The icon name to use for the primary icon for the entry.

Flags: Read / Write

Default value: NULL

Since: 2.16

The “primary-icon-pixbuf” property

“primary-icon-pixbuf” GdkPixbuf *

A pixbuf to use as the primary icon for the entry.

Flags: Read / Write

Since: 2.16

The “primary-icon-sensitive” property

“primary-icon-sensitive” gboolean

Whether the primary icon is sensitive.

An insensitive icon appears grayed out. GTK+ does not emit the “icon-press” and “icon-release” signals and does not allow DND from insensitive icons.

An icon should be set insensitive if the action that would trigger when clicked is currently not available.

Flags: Read / Write

Default value: TRUE

Since: 2.16

The “primary-icon-stock” property

“primary-icon-stock” gchar *

The stock id to use for the primary icon for the entry.

GtkEntry:primary-icon-stock has been deprecated since version 3.10 and should not be used in newly-written code.

Use “primary-icon-name” instead.

Flags: Read / Write

Default value: NULL

Since: 2.16

The “primary-icon-storage-type” property

“primary-icon-storage-type” GtkImageType

The representation which is used for the primary icon of the entry.

Flags: Read

Default value: GTK_IMAGE_EMPTY

Since: 2.16

The “primary-icon-tooltip-markup” property

“primary-icon-tooltip-markup” gchar *

The contents of the tooltip on the primary icon, which is marked up with the Pango text markup language.

Also see gtk_entry_set_icon_tooltip_markup().

Flags: Read / Write

Default value: NULL

Since: 2.16

The “primary-icon-tooltip-text” property

“primary-icon-tooltip-text” gchar *

The contents of the tooltip on the primary icon.

Also see gtk_entry_set_icon_tooltip_text().

Flags: Read / Write

Default value: NULL

Since: 2.16

The “progress-fraction” property

“progress-fraction” gdouble

The current fraction of the task that's been completed.

Flags: Read / Write

Allowed values: [0,1]

Default value: 0

Since: 2.16

The “progress-pulse-step” property

“progress-pulse-step” gdouble

The fraction of total entry width to move the progress bouncing block for each call to gtk_entry_progress_pulse().

Flags: Read / Write

Allowed values: [0,1]

Default value: 0.1

Since: 2.16

The “scroll-offset” property

“scroll-offset” gint

Number of pixels of the entry scrolled off the screen to the left.

Flags: Read

Allowed values: >= 0

Default value: 0

The “secondary-icon-activatable” property

“secondary-icon-activatable” gboolean

Whether the secondary icon is activatable.

GTK+ emits the “icon-press” and “icon-release” signals only on sensitive, activatable icons.

Sensitive, but non-activatable icons can be used for purely informational purposes.

Flags: Read / Write

Default value: TRUE

Since: 2.16

The “secondary-icon-gicon” property

“secondary-icon-gicon” GIcon *

The GIcon to use for the secondary icon for the entry.

Flags: Read / Write

Since: 2.16

The “secondary-icon-name” property

“secondary-icon-name” gchar *

The icon name to use for the secondary icon for the entry.

Flags: Read / Write

Default value: NULL

Since: 2.16

The “secondary-icon-pixbuf” property

“secondary-icon-pixbuf” GdkPixbuf *

An pixbuf to use as the secondary icon for the entry.

Flags: Read / Write

Since: 2.16

The “secondary-icon-sensitive” property

“secondary-icon-sensitive” gboolean

Whether the secondary icon is sensitive.

An insensitive icon appears grayed out. GTK+ does not emit the “icon-press” and “icon-release” signals and

does not allow DND from insensitive icons.

An icon should be set insensitive if the action that would trigger when clicked is currently not available.

Flags: Read / Write

Default value: TRUE

Since: 2.16

The `“secondary-icon-stock”` property

```
“secondary-icon-stock”     gchar *
```

The stock id to use for the secondary icon for the entry.

`GtkEntry:secondary-icon-stock` has been deprecated since version 3.10 and should not be used in newly-written code.

Use “secondary-icon-name” instead.

Flags: Read / Write

Default value: NULL

Since: 2.16

The `“secondary-icon-storage-type”` property

```
“secondary-icon-storage-type” GtkImageType
```

The representation which is used for the secondary icon of the entry.

Flags: Read

Default value: GTK_IMAGE_EMPTY

Since: 2.16

The `“secondary-icon-tooltip-markup”` property

```
“secondary-icon-tooltip-markup” gchar *
```

The contents of the tooltip on the secondary icon, which is marked up with the Pango text markup language.

Also see `gtk_entry_set_icon_tooltip_markup()`.

Flags: Read / Write

Default value: NULL

Since: 2.16

The “secondary-icon-tooltip-text” property

“secondary-icon-tooltip-text” gchar *

The contents of the tooltip on the secondary icon.

Also see gtk_entry_set_icon_tooltip_text().

Flags: Read / Write

Default value: NULL

Since: 2.16

The “selection-bound” property

“selection-bound” gint

The position of the opposite end of the selection from the cursor in chars.

Flags: Read

Allowed values: [0,65535]

Default value: 0

The “shadow-type” property

“shadow-type” GtkShadowType

Which kind of shadow to draw around the entry when “has-frame” is set to TRUE.

GtkEntry:shadow-type has been deprecated since version 3.20 and should not be used in newly-written code.

Use CSS to determine the style of the border; the value of this style property is ignored.

Flags: Read / Write

Default value: GTK_SHADOW_IN

Since: 2.12

The “show-emoji-icon” property

“show-emoji-icon” gboolean

Whether to show an icon for Emoji.

Flags: Read / Write

Default value: FALSE

The “tabs” property

“tabs” PangoTabArray *

A list of tabstop locations to apply to the text of the entry.

Flags: Read / Write

The “text” property

“text” gchar *

The contents of the entry.

Flags: Read / Write

Default value: ""

The “text-length” property

“text-length” guint

The length of the text in the GtkEntry.

Flags: Read

Allowed values: <= 65535

Default value: 0

Since: 2.14

The “truncate-multiline” property

“truncate-multiline” gboolean

When TRUE, pasted multi-line text is truncated to the first line.

Flags: Read / Write

Default value: FALSE

Since: 2.10

The “visibility” property

“visibility” gboolean

FALSE displays the "invisible char" instead of the actual text (password mode).

Flags: Read / Write

Default value: TRUE

The “width-chars” property

“width-chars” gint

Number of characters to leave space for in the entry.

Flags: Read / Write

Allowed values: >= -1

Default value: -1

The “xalign” property

“xalign” gfloat

The horizontal alignment, from 0 (left) to 1 (right). Reversed for RTL layouts.

Flags: Read / Write

Allowed values: [0,1]

Default value: 0

Since: 2.4

Style Property Details

The “icon-prelight” style property

“icon-prelight” gboolean

The prelight style property determines whether activatable icons prelight on mouseover.

GtkEntry:icon-prelight has been deprecated since version 3.20 and should not be used in newly-written code.

Use CSS to control the appearance of prelighted icons; the value of this style property is ignored.

Flags: Read

Default value: TRUE

Since: 2.16

The “inner-border” style property

“inner-border” GtkBorder *

Sets the text area's border between the text and the frame.

GtkEntry:inner-border has been deprecated since version 3.4 and should not be used in newly-written code.

Use the standard border and padding CSS properties (through objects like GtkStyleContext and GtkCssProvider); the value of this style property is ignored.

Flags: Read

Since: 2.10

The “invisible-char” style property

“invisible-char” guint

The invisible character is used when masking entry contents (in \"password mode\")"). When it is not explicitly set with the “invisible-char” property, GTK+ determines the character to use from a list of possible candidates, depending on availability in the current font.

This style property allows the theme to prepend a character to the list of candidates.

Flags: Read

Default value: 0

Since: 2.18

The “progress-border” style property

“progress-border” GtkBorder *

The border around the progress bar in the entry.

GtkEntry:progress-border has been deprecated since version 3.4 and should not be used in newly-written code.

Use the standard margin CSS property (through objects like GtkStyleContext and GtkCssProvider); the value of this style property is ignored.

Flags: Read

Since: 2.16

Signal Details

The “activate” signal

```
void
user_function (GtkEntry *entry,
```

```
                gpointer  user_data)
```

The ::activate signal is emitted when the user hits the Enter key.

While this signal is used as a keybinding signal, it is also commonly used by applications to intercept activation of entries.

The default bindings for this signal are all forms of the Enter key.

Parameters

entry	The entry on which the signal is emitted
user_data	user data set when the signal handler was connected.

Flags: Action

The `"backspace"` signal

```
void
user_function (GtkEntry *entry,
               gpointer  user_data)
```

The ::backspace signal is a keybinding signal which gets emitted when the user asks for it.

The default bindings for this signal are Backspace and Shift-Backspace.

Parameters

entry	the object which received the signal
user_data	user data set when the signal handler was connected.

Flags: Action

The `"copy-clipboard"` signal

```
void
user_function (GtkEntry *entry,
               gpointer  user_data)
```

The ::copy-clipboard signal is a keybinding signal which gets emitted to copy the selection to the clipboard.

The default bindings for this signal are Ctrl-c and Ctrl-Insert.

Parameters

entry	the object which received the signal
user_data	user data set when the signal handler was connected.

Flags: Action

The “cut-clipboard” signal

```
void
user_function (GtkEntry *entry,
               gpointer  user_data)
```

The ::cut-clipboard signal is a keybinding signal which gets emitted to cut the selection to the clipboard.

The default bindings for this signal are Ctrl-x and Shift-Delete.

Parameters

entry	the object which received the signal
user_data	user data set when the signal handler was connected.

Flags: Action

The “delete-from-cursor” signal

```
void
user_function (GtkEntry     *entry,
               GtkDeleteType type,
               gint          count,
               gpointer      user_data)
```

The ::delete-from-cursor signal is a keybinding signal which gets emitted when the user initiates a text deletion.

If the `type` is `GTK_DELETE_CHARS`, GTK+ deletes the selection if there is one, otherwise it deletes the requested number of characters.

The default bindings for this signal are Delete for deleting a character and Ctrl-Delete for deleting a word.

Parameters

entry	the object which received the signal
type	the granularity of the deletion, as a GtkDeleteType
count	the number of `type` units to delete
user_data	user data set when the signal handler was connected.

Flags: Action

The “icon-press” signal

```
void
user_function (GtkEntry            *entry,
               GtkEntryIconPosition icon_pos,
               GdkEvent            *event,
               gpointer             user_data)
```

The ::icon-press signal is emitted when an activatable icon is clicked.

Parameters

entry	The entry on which the signal is emitted	
icon_pos	The position of the clicked icon	
event	the button press event.	[type Gdk.EventButton]
user_data	user data set when the signal handler was connected.	

Flags: Run Last

Since: 2.16

The `"icon-release"` signal

```
void
user_function (GtkEntry            *entry,
               GtkEntryIconPosition icon_pos,
               GdkEvent            *event,
               gpointer             user_data)
```

The ::icon-release signal is emitted on the button release from a mouse click over an activatable icon.

Parameters

entry	The entry on which the signal is emitted	
icon_pos	The position of the clicked icon	
event	the button release event.	[type Gdk.EventButton]
user_data	user data set when the signal handler was connected.	

Flags: Run Last

Since: 2.16

The `"insert-at-cursor"` signal

```
void
user_function (GtkEntry *entry,
               gchar    *string,
               gpointer  user_data)
```

The ::insert-at-cursor signal is a keybinding signal which gets emitted when the user initiates the insertion of a fixed string at the cursor.

This signal has no default bindings.

Parameters

entry	the object which received the signal

string	the string to insert
user_data	user data set when the signal handler was connected.

Flags: Action

The `“insert-emoji”` signal

```
void
user_function (GtkEntry *entry,
               gpointer  user_data)
```

The ::insert-emoji signal is a keybinding signal which gets emitted to present the Emoji chooser for the `entry` .

The default bindings for this signal are Ctrl-. and Ctrl-;

Parameters

entry	the object which received the signal
user_data	user data set when the signal handler was connected.

Flags: Action

Since: 3.22.27

The `“move-cursor”` signal

```
void
user_function (GtkEntry       *entry,
               GtkMovementStep step,
               gint            count,
               gboolean        extend_selection,
               gpointer        user_data)
```

The ::move-cursor signal is a keybinding signal which gets emitted when the user initiates a cursor movement. If the cursor is not visible in `entry` , this signal causes the viewport to be moved instead.

Applications should not connect to it, but may emit it with `g_signal_emit_by_name()` if they need to control the cursor programmatically.

The default bindings for this signal come in two variants, the variant with the Shift modifier extends the selection, the variant without the Shift modifer does not. There are too many key combinations to list them all here.

- Arrow keys move by individual characters/lines
- Ctrl-arrow key combinations move by words/paragraphs
- Home/End keys move to the ends of the buffer

Parameters

entry	the object which received the signal

step	the granularity of the move, as a GtkMovementStep
count	the number of `step` units to move
extend_selection	`TRUE` if the move should extend the selection
user_data	user data set when the signal handler was connected.

Flags: Action

The “paste-clipboard” signal

```
void
user_function (GtkEntry *entry,
               gpointer  user_data)
```

The ::paste-clipboard signal is a keybinding signal which gets emitted to paste the contents of the clipboard into the text view.

The default bindings for this signal are Ctrl-v and Shift-Insert.

Parameters

entry	the object which received the signal
user_data	user data set when the signal handler was connected.

Flags: Action

The “populate-popup” signal

```
void
user_function (GtkEntry  *entry,
               GtkWidget *widget,
               gpointer   user_data)
```

The ::populate-popup signal gets emitted before showing the context menu of the entry.

If you need to add items to the context menu, connect to this signal and append your items to the `widget` , which will be a GtkMenu in this case.

If “populate-all” is `TRUE`, this signal will also be emitted to populate touch popups. In this case, `widget` will be a different container, e.g. a GtkToolbar. The signal handler should not make assumptions about the type of `widget` .

Parameters

entry	The entry on which the signal is emitted
widget	the container that is being populated
user_data	user data set when the signal handler was connected.

Flags: Run Last

The “preedit-changed” signal

```
void
user_function (GtkEntry *entry,
               gchar    *preedit,
               gpointer  user_data)
```

If an input method is used, the typed text will not immediately be committed to the buffer. So if you are interested in the text, connect to this signal.

Parameters

entry	the object which received the signal
preedit	the current preedit string
user_data	user data set when the signal handler was connected.

Flags: Action

Since: 2.20

The “toggle-overwrite” signal

```
void
user_function (GtkEntry *entry,
               gpointer  user_data)
```

The ::toggle-overwrite signal is a keybinding signal which gets emitted to toggle the overwrite mode of the entry.

The default bindings for this signal is Insert.

Parameters

entry	the object which received the signal
user_data	user data set when the signal handler was connected.

Flags: Action

See Also

GtkTextView, GtkEntryCompletion

GtkEntryBuffer

GtkEntryBuffer — Text buffer for GtkEntry

Functions

GtkEntryBuffer *	gtk_entry_buffer_new ()
const gchar *	gtk_entry_buffer_get_text ()
void	gtk_entry_buffer_set_text ()
gsize	gtk_entry_buffer_get_bytes ()
guint	gtk_entry_buffer_get_length ()
gint	gtk_entry_buffer_get_max_length ()
void	gtk_entry_buffer_set_max_length ()
guint	gtk_entry_buffer_insert_text ()
guint	gtk_entry_buffer_delete_text ()
void	gtk_entry_buffer_emit_deleted_text ()
void	gtk_entry_buffer_emit_inserted_text ()

Properties

guint	length	Read
gint	max-length	Read / Write
gchar *	text	Read / Write

Signals

void	deleted-text	Run First
void	inserted-text	Run First

Types and Values

struct	GtkEntryBuffer

Object Hierarchy

```
GObject
└── GtkEntryBuffer
```

Includes

```
#include <gtk/gtk.h>
```

Description

The GtkEntryBuffer class contains the actual text displayed in a GtkEntry widget.

A single GtkEntryBuffer object can be shared by multiple GtkEntry widgets which will then share the same text content, but not the cursor position, visibility attributes, icon etc.

GtkEntryBuffer may be derived from. Such a derived class might allow text to be stored in an alternate location,

such as non-pageable memory, useful in the case of important passwords. Or a derived class could integrate with an application’s concept of undo/redo.

Functions

gtk_entry_buffer_new ()

```
GtkEntryBuffer *
gtk_entry_buffer_new (const gchar *initial_chars,
                      gint n_initial_chars);
```

Create a new GtkEntryBuffer object.

Optionally, specify initial text to set in the buffer.

Parameters

initial_chars	initial buffer text, or `NULL`.	[allow-none]
n_initial_chars	number of characters in `initial_chars` , or -1	

Returns

A new GtkEntryBuffer object.

Since: 2.18

---

gtk_entry_buffer_get_text ()

```
const gchar *
gtk_entry_buffer_get_text (GtkEntryBuffer *buffer);
```

Retrieves the contents of the buffer.

The memory pointer returned by this call will not change unless this object emits a signal, or is finalized.

Parameters

buffer	a GtkEntryBuffer

Returns

a pointer to the contents of the widget as a string. This string points to internally allocated storage in the buffer and must not be freed, modified or stored.

Since: 2.18

---

gtk_entry_buffer_set_text ()

```
void
gtk_entry_buffer_set_text (GtkEntryBuffer *buffer,
                           const gchar *chars,
                           gint n_chars);
```

Sets the text in the buffer.

This is roughly equivalent to calling `gtk_entry_buffer_delete_text()` and `gtk_entry_buffer_insert_text()`.

Note that `n_chars` is in characters, not in bytes.

Parameters

buffer	a GtkEntryBuffer
chars	the new text
n_chars	the number of characters in `text` , or -1

Since: 2.18

---

gtk_entry_buffer_get_bytes ()

```
gsize
gtk_entry_buffer_get_bytes (GtkEntryBuffer *buffer);
```

Retrieves the length in bytes of the buffer. See `gtk_entry_buffer_get_length()`.

Parameters

buffer	a GtkEntryBuffer

Returns

The byte length of the buffer.

Since: 2.18

---

gtk_entry_buffer_get_length ()

```
guint
gtk_entry_buffer_get_length (GtkEntryBuffer *buffer);
```

Retrieves the length in characters of the buffer.

Parameters

buffer	a GtkEntryBuffer

Returns

The number of characters in the buffer.

Since: 2.18

---

gtk_entry_buffer_get_max_length ()

```
gint
gtk_entry_buffer_get_max_length (GtkEntryBuffer *buffer);
```

Retrieves the maximum allowed length of the text in `buffer` . See `gtk_entry_buffer_set_max_length()`.

Parameters

buffer	a GtkEntryBuffer

Returns

the maximum allowed number of characters in GtkEntryBuffer, or 0 if there is no maximum.

Since: 2.18

---

gtk_entry_buffer_set_max_length ()

```
void
gtk_entry_buffer_set_max_length (GtkEntryBuffer *buffer,
                                 gint max_length);
```

Sets the maximum allowed length of the contents of the buffer. If the current contents are longer than the given length, then they will be truncated to fit.

Parameters

buffer	a GtkEntryBuffer
max_length	the maximum length of the entry buffer, or 0 for no maximum. (other than the maximum length of entries.) The value passed in will be clamped to the range 0-65536.

Since: 2.18

---

gtk_entry_buffer_insert_text ()

```
guint
gtk_entry_buffer_insert_text (GtkEntryBuffer *buffer,
                              guint position,
                              const gchar *chars,
```

```
                                gint n_chars);
```

Inserts `n_chars` characters of `chars` into the contents of the buffer, at position `position` .

If `n_chars` is negative, then characters from chars will be inserted until a null-terminator is found. If `position` or `n_chars` are out of bounds, or the maximum buffer text length is exceeded, then they are coerced to sane values.

Note that the position and length are in characters, not in bytes.

Parameters

buffer	a GtkEntryBuffer
position	the position at which to insert text.
chars	the text to insert into the buffer.
n_chars	the length of the text in characters, or -1

Returns

The number of characters actually inserted.

Since: 2.18

---

gtk_entry_buffer_delete_text ()

```
guint
gtk_entry_buffer_delete_text (GtkEntryBuffer *buffer,
                              guint position,
                              gint n_chars);
```

Deletes a sequence of characters from the buffer. `n_chars` characters are deleted starting at `position` . If `n_chars` is negative, then all characters until the end of the text are deleted.

If `position` or `n_chars` are out of bounds, then they are coerced to sane values.

Note that the positions are specified in characters, not bytes.

Parameters

buffer	a GtkEntryBuffer
position	position at which to delete text
n_chars	number of characters to delete

Returns

The number of characters deleted.

Since: 2.18

---

gtk_entry_buffer_emit_deleted_text ()

```
void
gtk_entry_buffer_emit_deleted_text (GtkEntryBuffer *buffer,
                                    guint position,
                                    guint n_chars);
```

Used when subclassing GtkEntryBuffer

Parameters

buffer	a GtkEntryBuffer
position	position at which text was deleted
n_chars	number of characters deleted

Since: 2.18

---

gtk_entry_buffer_emit_inserted_text ()

```
void
gtk_entry_buffer_emit_inserted_text (GtkEntryBuffer *buffer,
                                     guint position,
                                     const gchar *chars,
                                     guint n_chars);
```

Used when subclassing GtkEntryBuffer

Parameters

buffer	a GtkEntryBuffer
position	position at which text was inserted
chars	text that was inserted
n_chars	number of characters inserted

Since: 2.18

Types and Values

struct GtkEntryBuffer

```
struct GtkEntryBuffer;
```

Property Details

The “length” property

```
  “length”                   guint
```

The length (in characters) of the text in buffer.

Flags: Read

Allowed values: <= 65535

Default value: 0

Since: 2.18

The “max-length” property

```
  “max-length”               gint
```

The maximum length (in characters) of the text in the buffer.

Flags: Read / Write

Allowed values: [0,65535]

Default value: 0

Since: 2.18

The “text” property

```
  “text”                     gchar *
```

The contents of the buffer.

Flags: Read / Write

Default value: ""

Since: 2.18

Signal Details

The “deleted-text” signal

```
void
user_function (GtkEntryBuffer *buffer,
               guint           position,
               guint           n_chars,
               gpointer        user_data)
```

This signal is emitted after text is deleted from the buffer.

Parameters

buffer	a GtkEntryBuffer
position	the position the text was deleted at.
n_chars	The number of characters that were deleted.
user_data	user data set when the signal

handler was connected.

Flags: Run First

Since: 2.18

The `"inserted-text"` signal

```
void
user_function (GtkEntryBuffer *buffer,
               guint           position,
               gchar          *chars,
               guint           n_chars,
               gpointer        user_data)
```

This signal is emitted after text is inserted into the buffer.

Parameters

buffer	a GtkEntryBuffer
position	the position the text was inserted at.
chars	The text that was inserted.
n_chars	The number of characters that were inserted.
user_data	user data set when the signal handler was connected.

Flags: Run First

Since: 2.18

GtkEntryCompletion

GtkEntryCompletion — Completion functionality for GtkEntry

Functions

gboolean	(*GtkEntryCompletionMatchFunc) ()
GtkEntryCompletion *	gtk_entry_completion_new ()
GtkEntryCompletion *	gtk_entry_completion_new_with_area ()
GtkWidget *	gtk_entry_completion_get_entry ()
void	gtk_entry_completion_set_model ()
GtkTreeModel *	gtk_entry_completion_get_model ()
void	gtk_entry_completion_set_match_func ()
void	gtk_entry_completion_set_minimum_key_length ()
gint	gtk_entry_completion_get_minimum_key_length ()
gchar *	gtk_entry_completion_compute_prefix ()
void	gtk_entry_completion_complete ()
const gchar *	gtk_entry_completion_get_completion_prefix ()
void	gtk_entry_completion_insert_prefix ()
void	gtk_entry_completion_insert_action_text ()
void	gtk_entry_completion_insert_action_markup ()

void	gtk_entry_completion_delete_action ()
void	gtk_entry_completion_set_text_column ()
gint	gtk_entry_completion_get_text_column ()
void	gtk_entry_completion_set_inline_completion ()
gboolean	gtk_entry_completion_get_inline_completion ()
void	gtk_entry_completion_set_inline_selection ()
gboolean	gtk_entry_completion_get_inline_selection ()
void	gtk_entry_completion_set_popup_completion ()
gboolean	gtk_entry_completion_get_popup_completion ()
void	gtk_entry_completion_set_popup_set_width ()
gboolean	gtk_entry_completion_get_popup_set_width ()
void	gtk_entry_completion_set_popup_single_match ()
gboolean	gtk_entry_completion_get_popup_single_match ()

Properties

GtkCellArea *	cell-area	Read / Write / Construct Only
gboolean	inline-completion	Read / Write
gboolean	inline-selection	Read / Write
gint	minimum-key-length	Read / Write
GtkTreeModel *	model	Read / Write
gboolean	popup-completion	Read / Write
gboolean	popup-set-width	Read / Write
gboolean	popup-single-match	Read / Write
gint	text-column	Read / Write

Signals

void	action-activated	Run Last
gboolean	cursor-on-match	Run Last
gboolean	insert-prefix	Run Last
gboolean	match-selected	Run Last
void	no-matches	Run Last

Types and Values

struct	GtkEntryCompletion

Object Hierarchy

```
GObject
└── GtkEntryCompletion
```

Implemented Interfaces

GtkEntryCompletion implements GtkCellLayout and GtkBuildable.

Includes

```
#include <gtk/gtk.h>
```

Description

GtkEntryCompletion is an auxiliary object to be used in conjunction with GtkEntry to provide the completion functionality. It implements the GtkCellLayout interface, to allow the user to add extra cells to the GtkTreeView with completion matches.

“Completion functionality” means that when the user modifies the text in the entry, GtkEntryCompletion checks which rows in the model match the current content of the entry, and displays a list of matches. By default, the matching is done by comparing the entry text case-insensitively against the text column of the model (see `gtk_entry_completion_set_text_column()`), but this can be overridden with a custom match function (see `gtk_entry_completion_set_match_func()`).

When the user selects a completion, the content of the entry is updated. By default, the content of the entry is replaced by the text column of the model, but this can be overridden by connecting to the “match-selected” signal and updating the entry in the signal handler. Note that you should return `TRUE` from the signal handler to suppress the default behaviour.

To add completion functionality to an entry, use `gtk_entry_set_completion()`.

In addition to regular completion matches, which will be inserted into the entry when they are selected, GtkEntryCompletion also allows to display “actions” in the popup window. Their appearance is similar to menuitems, to differentiate them clearly from completion strings. When an action is selected, the “action-activated” signal is emitted.

GtkEntryCompletion uses a GtkTreeModelFilter model to represent the subset of the entire model that is currently matching. While the GtkEntryCompletion signals “match-selected” and “cursor-on-match” take the original model and an iter pointing to that model as arguments, other callbacks and signals (such as GtkCellLayoutDataFuncs or “apply-attributes”) will generally take the filter model as argument. As long as you are only calling `gtk_tree_model_get()`, this will make no difference to you. If for some reason, you need the original model, use `gtk_tree_model_filter_get_model()`. Don’t forget to use `gtk_tree_model_filter_convert_iter_to_child_iter()` to obtain a matching iter.

Functions

GtkEntryCompletionMatchFunc ()

```
gboolean
(*GtkEntryCompletionMatchFunc) (GtkEntryCompletion *completion,
                                const gchar *key,
                                GtkTreeIter *iter,
                                gpointer user_data);
```

A function which decides whether the row indicated by `iter` matches a given `key` , and should be displayed as a possible completion for `key` . Note that `key` is normalized and case-folded (see `g_utf8_normalize()` and `g_utf8_casefold()`). If this is not appropriate, match functions have access to the unmodified key via `gtk_entry_get_text (GTK_ENTRY (gtk_entry_completion_get_entry()))`.

Parameters

completion	the GtkEntryCompletion
key	the string to match, normalized and case-folded
iter	a GtkTreeIter indicating the row to match
user_data	user data given to `gtk_entry_completion_set_match_func()`

Returns

`TRUE` if `iter` should be displayed as a possible completion for `key`

---

gtk_entry_completion_new ()

```
GtkEntryCompletion *
gtk_entry_completion_new (void);
```

Creates a new GtkEntryCompletion object.

Returns

A newly created GtkEntryCompletion object

Since: 2.4

---

gtk_entry_completion_new_with_area ()

```
GtkEntryCompletion *
gtk_entry_completion_new_with_area (GtkCellArea *area);
```

Creates a new GtkEntryCompletion object using the specified `area` to layout cells in the underlying GtkTreeViewColumn for the drop-down menu.

Parameters

area	the GtkCellArea used to layout cells

Returns

A newly created GtkEntryCompletion object

Since: 3.0

---

gtk_entry_completion_get_entry ()

```
GtkWidget *
gtk_entry_completion_get_entry (GtkEntryCompletion *completion);
```

Gets the entry `completion` has been attached to.

Parameters

completion	a GtkEntryCompletion	

Returns

The entry `completion` has been attached to.

[transfer none]

Since: 2.4

---

gtk_entry_completion_set_model ()

```
void
gtk_entry_completion_set_model (GtkEntryCompletion *completion,
                                GtkTreeModel *model);
```

Sets the model for a GtkEntryCompletion. If `completion` already has a model set, it will remove it before setting the new model. If model is `NULL`, then it will unset the model.

Parameters

completion	a GtkEntryCompletion	
model	the GtkTreeModel.	[allow-none]

Since: 2.4

---

gtk_entry_completion_get_model ()

```
GtkTreeModel *
gtk_entry_completion_get_model (GtkEntryCompletion *completion);
```

Returns the model the GtkEntryCompletion is using as data source. Returns `NULL` if the model is unset.

Parameters

completion	a GtkEntryCompletion	

Returns

A GtkTreeModel, or `NULL` if none is currently being used.

[nullable][transfer none]

Since: 2.4

---

gtk_entry_completion_set_match_func ()

```
void
gtk_entry_completion_set_match_func (GtkEntryCompletion *completion,
                                     GtkEntryCompletionMatchFunc func,
                                     gpointer func_data,
                                     GDestroyNotify func_notify);
```

Sets the match function for `completion` to be `func` . The match function is used to determine if a row should or should not be in the completion list.

Parameters

completion	a GtkEntryCompletion
func	the GtkEntryCompletionMatchFunc to use
func_data	user data for `func`
func_notify	destroy notify for `func_data` .

Since: 2.4

---

gtk_entry_completion_set_minimum_key_length ()

```
void
gtk_entry_completion_set_minimum_key_length
                                (GtkEntryCompletion *completion,
                                 gint length);
```

Requires the length of the search key for `completion` to be at least `length` . This is useful for long lists, where completing using a small key takes a lot of time and will come up with meaningless results anyway (ie, a too large dataset).

Parameters

completion	a GtkEntryCompletion
length	the minimum length of the key in order to start completing

Since: 2.4

---

gtk_entry_completion_get_minimum_key_length ()

```
gint
gtk_entry_completion_get_minimum_key_length
                                (GtkEntryCompletion *completion);
```

Returns the minimum key length as set for `completion` .

Parameters

completion	a GtkEntryCompletion

Returns

The currently used minimum key length

Since: 2.4

---

gtk_entry_completion_compute_prefix ()

```
gchar *
gtk_entry_completion_compute_prefix (GtkEntryCompletion *completion,
                                     const char *key);
```

Computes the common prefix that is shared by all rows in `completion` that start with `key` . If no row matches `key` , `NULL` will be returned. Note that a text column must have been set for this function to work, see `gtk_entry_completion_set_text_column()` for details.

Parameters

completion	the entry completion
key	The text to complete for

Returns

The common prefix all rows starting with `key` or `NULL` if no row matches `key` .

[nullable][transfer full]

Since: 3.4

---

gtk_entry_completion_complete ()

```
void
gtk_entry_completion_complete (GtkEntryCompletion *completion);
```

Requests a completion operation, or in other words a refiltering of the current list with completions, using the current key. The completion list view will be updated accordingly.

Parameters

completion	a GtkEntryCompletion

Since: 2.4

---

gtk_entry_completion_get_completion_prefix ()

```
const gchar *
gtk_entry_completion_get_completion_prefix
                                (GtkEntryCompletion *completion);
```

Get the original text entered by the user that triggered the completion or `NULL` if there’s no completion ongoing.

Parameters

completion	a GtkEntryCompletion

Returns

the prefix for the current completion

Since: 2.12

---

gtk_entry_completion_insert_prefix ()

```
void
gtk_entry_completion_insert_prefix (GtkEntryCompletion *completion);
```

Requests a prefix insertion.

Parameters

completion	a GtkEntryCompletion

Since: 2.6

---

gtk_entry_completion_insert_action_text ()

```
void
gtk_entry_completion_insert_action_text
                                (GtkEntryCompletion *completion,
                                 gint index_,
                                 const gchar *text);
```

Inserts an action in `completion` 's action item list at position `index_` with text `text` . If you want the action item to have markup, use `gtk_entry_completion_insert_action_markup()`.

Note that `index_` is a relative position in the list of actions and the position of an action can change when deleting a different action.

Parameters

completion	a GtkEntryCompletion
index_	the index of the item to insert
text	text of the item to insert

Since: 2.4

---

gtk_entry_completion_insert_action_markup ()

```
void
```

```
gtk_entry_completion_insert_action_markup
                                (GtkEntryCompletion *completion,
                                 gint index_,
                                 const gchar *markup);
```

Inserts an action in `completion` 's action item list at position `index_` with markup `markup` .

Parameters

completion	a GtkEntryCompletion
index_	the index of the item to insert
markup	markup of the item to insert

Since: 2.4

---

gtk_entry_completion_delete_action ()

```
void
gtk_entry_completion_delete_action (GtkEntryCompletion *completion,
                                    gint index_);
```

Deletes the action at `index_` from `completion` 's action list.

Note that `index_` is a relative position and the position of an action may have changed since it was inserted.

Parameters

completion	a GtkEntryCompletion
index_	the index of the item to delete

Since: 2.4

---

gtk_entry_completion_set_text_column ()

```
void
gtk_entry_completion_set_text_column (GtkEntryCompletion *completion,
                                      gint column);
```

Convenience function for setting up the most used case of this code: a completion list with just strings. This function will set up `completion` to have a list displaying all (and just) strings in the completion list, and to get those strings from `column` in the model of `completion` .

This functions creates and adds a GtkCellRendererText for the selected column. If you need to set the text column, but don't want the cell renderer, use `g_object_set()` to set the "text-column" property directly.

Parameters

completion	a GtkEntryCompletion
column	the column in the model of `completion` to get strings from

Since: 2.4

---

gtk_entry_completion_get_text_column ()

```
gint
gtk_entry_completion_get_text_column (GtkEntryCompletion *completion);
```

Returns the column in the model of `completion` to get strings from.

Parameters

completion	a GtkEntryCompletion

Returns

the column containing the strings

Since: 2.6

---

gtk_entry_completion_set_inline_completion ()

```
void
gtk_entry_completion_set_inline_completion
                                (GtkEntryCompletion *completion,
                                 gboolean inline_completion);
```

Sets whether the common prefix of the possible completions should be automatically inserted in the entry.

Parameters

completion	a GtkEntryCompletion
inline_completion	`TRUE` to do inline completion

Since: 2.6

---

gtk_entry_completion_get_inline_completion ()

```
gboolean
gtk_entry_completion_get_inline_completion
                                (GtkEntryCompletion *completion);
```

Returns whether the common prefix of the possible completions should be automatically inserted in the entry.

Parameters

completion	a GtkEntryCompletion

Returns

`TRUE` if inline completion is turned on

Since: 2.6

---

gtk_entry_completion_set_inline_selection ()

```
void
gtk_entry_completion_set_inline_selection
                                (GtkEntryCompletion *completion,
                                 gboolean inline_selection);
```

Sets whether it is possible to cycle through the possible completions inside the entry.

Parameters

completion	a GtkEntryCompletion
inline_selection	TRUE to do inline selection

Since: 2.12

---

gtk_entry_completion_get_inline_selection ()

```
gboolean
gtk_entry_completion_get_inline_selection
                                (GtkEntryCompletion *completion);
```

Returns TRUE if inline-selection mode is turned on.

Parameters

completion	a GtkEntryCompletion

Returns

TRUE if inline-selection mode is on

Since: 2.12

---

gtk_entry_completion_set_popup_completion ()

```
void
gtk_entry_completion_set_popup_completion
                                (GtkEntryCompletion *completion,
                                 gboolean popup_completion);
```

Sets whether the completions should be presented in a popup window.

Parameters

completion	a GtkEntryCompletion

popup_completion	TRUE to do popup completion

Since: 2.6

---

gtk_entry_completion_get_popup_completion ()

```
gboolean
gtk_entry_completion_get_popup_completion
                                (GtkEntryCompletion *completion);
```

Returns whether the completions should be presented in a popup window.

Parameters

completion	a GtkEntryCompletion

Returns

TRUE if popup completion is turned on

Since: 2.6

---

gtk_entry_completion_set_popup_set_width ()

```
void
gtk_entry_completion_set_popup_set_width
                                (GtkEntryCompletion *completion,
                                 gboolean popup_set_width);
```

Sets whether the completion popup window will be resized to be the same width as the entry.

Parameters

completion	a GtkEntryCompletion
popup_set_width	TRUE to make the width of the popup the same as the entry

Since: 2.8

---

gtk_entry_completion_get_popup_set_width ()

```
gboolean
gtk_entry_completion_get_popup_set_width
                                (GtkEntryCompletion *completion);
```

Returns whether the completion popup window will be resized to the width of the entry.

Parameters

completion	a GtkEntryCompletion

Returns

TRUE if the popup window will be resized to the width of the entry

Since: 2.8

---

gtk_entry_completion_set_popup_single_match ()

```
void
gtk_entry_completion_set_popup_single_match
                                (GtkEntryCompletion *completion,
                                 gboolean popup_single_match);
```

Sets whether the completion popup window will appear even if there is only a single match. You may want to set this to FALSE if you are using inline completion.

Parameters

completion	a GtkEntryCompletion
popup_single_match	TRUE if the popup should appear even for a single match

Since: 2.8

---

gtk_entry_completion_get_popup_single_match ()

```
gboolean
gtk_entry_completion_get_popup_single_match
                                (GtkEntryCompletion *completion);
```

Returns whether the completion popup window will appear even if there is only a single match.

Parameters

completion	a GtkEntryCompletion

Returns

TRUE if the popup window will appear regardless of the number of matches

Since: 2.8

Types and Values

struct GtkEntryCompletion

```
struct GtkEntryCompletion;
```

Property Details

The “cell-area” property

```
  “cell-area”                GtkCellArea *
```

The GtkCellArea used to layout cell renderers in the treeview column.

If no area is specified when creating the entry completion with `gtk_entry_completion_new_with_area()` a horizontally oriented GtkCellAreaBox will be used.

Flags: Read / Write / Construct Only

Since: 3.0

The “inline-completion” property

```
  “inline-completion”        gboolean
```

Determines whether the common prefix of the possible completions should be inserted automatically in the entry. Note that this requires text-column to be set, even if you are using a custom match function.

Flags: Read / Write

Default value: FALSE

Since: 2.6

The “inline-selection” property

```
  “inline-selection”         gboolean
```

Determines whether the possible completions on the popup will appear in the entry as you navigate through them.

Flags: Read / Write

Default value: FALSE

Since: 2.12

The “minimum-key-length” property

```
  “minimum-key-length”       gint
```

Minimum length of the search key in order to look up matches.

Flags: Read / Write

Allowed values: >= 0

Default value: 1

The “model” property

“model” GtkTreeModel *

The model to find matches in.

Flags: Read / Write

The “popup-completion” property

“popup-completion” gboolean

Determines whether the possible completions should be shown in a popup window.

Flags: Read / Write

Default value: TRUE

Since: 2.6

The “popup-set-width” property

“popup-set-width” gboolean

Determines whether the completions popup window will be resized to the width of the entry.

Flags: Read / Write

Default value: TRUE

Since: 2.8

The “popup-single-match” property

“popup-single-match” gboolean

Determines whether the completions popup window will shown for a single possible completion. You probably want to set this to FALSE if you are using inline completion.

Flags: Read / Write

Default value: TRUE

Since: 2.8

The “text-column” property

```
  “text-column”              gint
```

The column of the model containing the strings. Note that the strings must be UTF-8.

Flags: Read / Write

Allowed values: >= -1

Default value: -1

Since: 2.6

Signal Details

The “action-activated” signal

```
void
user_function (GtkEntryCompletion *widget,
               gint                index,
               gpointer            user_data)
```

Gets emitted when an action is activated.

Parameters

widget	the object which received the signal
index	the index of the activated action
user_data	user data set when the signal handler was connected.

Flags: Run Last

Since: 2.4

The “cursor-on-match” signal

```
gboolean
user_function (GtkEntryCompletion *widget,
               GtkTreeModel       *model,
               GtkTreeIter        *iter,
               gpointer            user_data)
```

Gets emitted when a match from the cursor is on a match of the list. The default behaviour is to replace the contents of the entry with the contents of the text column in the row pointed to by `iter` .

Note that `model` is the model that was passed to `gtk_entry_completion_set_model()`.

Parameters

widget	the object which received the signal
model	the GtkTreeModel containing the matches

iter	a GtkTreeIter positioned at the selected match
user_data	user data set when the signal handler was connected.

Returns

TRUE if the signal has been handled

Flags: Run Last

Since: 2.12

The “insert-prefix” signal

```
gboolean
user_function (GtkEntryCompletion *widget,
               gchar              *prefix,
               gpointer            user_data)
```

Gets emitted when the inline autocompletion is triggered. The default behaviour is to make the entry display the whole prefix and select the newly inserted part.

Applications may connect to this signal in order to insert only a smaller part of the prefix into the entry - e.g. the entry used in the GtkFileChooser inserts only the part of the prefix up to the next '/'.

Parameters

widget	the object which received the signal
prefix	the common prefix of all possible completions
user_data	user data set when the signal handler was connected.

Returns

TRUE if the signal has been handled

Flags: Run Last

Since: 2.6

The “match-selected” signal

```
gboolean
user_function (GtkEntryCompletion *widget,
               GtkTreeModel       *model,
               GtkTreeIter        *iter,
               gpointer            user_data)
```

Gets emitted when a match from the list is selected. The default behaviour is to replace the contents of the entry

with the contents of the text column in the row pointed to by `iter`.

Note that `model` is the model that was passed to `gtk_entry_completion_set_model()`.

Parameters

widget	the object which received the signal
model	the GtkTreeModel containing the matches
iter	a GtkTreeIter positioned at the selected match
user_data	user data set when the signal handler was connected.

Returns

`TRUE` if the signal has been handled

Flags: Run Last

Since: 2.4

The `"no-matches"` signal

```
void
user_function (GtkEntryCompletion *widget,
               gpointer            user_data)
```

Gets emitted when the filter model has zero number of rows in completion_complete method. (In other words when GtkEntryCompletion is out of suggestions)

Parameters

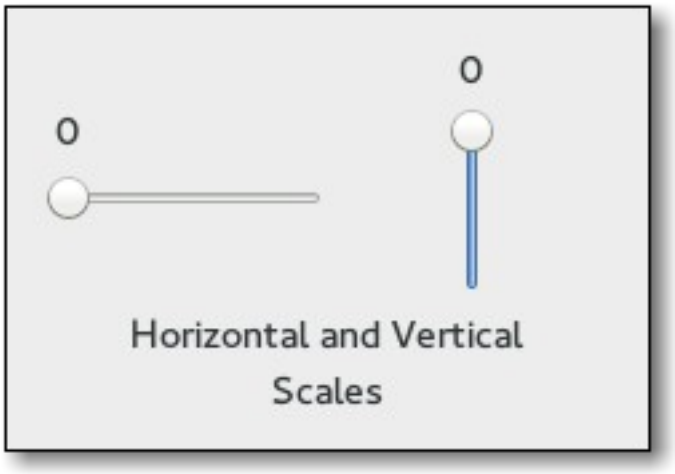
widget	the object which received the signal
user_data	user data set when the signal handler was connected.

Flags: Run Last

Since: 3.14

GtkScale

GtkScale — A slider widget for selecting a value from a range



Functions

GtkWidget *	gtk_scale_new ()
GtkWidget *	gtk_scale_new_with_range ()
void	gtk_scale_set_digits ()
void	gtk_scale_set_draw_value ()
void	gtk_scale_set_has_origin ()
void	gtk_scale_set_value_pos ()
gint	gtk_scale_get_digits ()
gboolean	gtk_scale_get_draw_value ()
gboolean	gtk_scale_get_has_origin ()
GtkPositionType	gtk_scale_get_value_pos ()
PangoLayout *	gtk_scale_get_layout ()
void	gtk_scale_get_layout_offsets ()
void	gtk_scale_add_mark ()
void	gtk_scale_clear_marks ()

Properties

gint	digits	Read / Write
gboolean	draw-value	Read / Write
gboolean	has-origin	Read / Write
GtkPositionType	value-pos	Read / Write

Style Properties

gint	slider-length	Read
gint	value-spacing	Read

Signals

gchar*	format-value	Run Last

Types and Values

struct	GtkScale

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkRange
            └── GtkScale
                ├── GtkHScale
                └── GtkVScale
```

Implemented Interfaces

GtkScale implements AtkImplementorIface, GtkBuildable and GtkOrientable.

Includes

```
#include <gtk/gtk.h>
```

Description

A GtkScale is a slider control used to select a numeric value. To use it, you'll probably want to investigate the methods on its base class, GtkRange, in addition to the methods for GtkScale itself. To set the value of a scale, you would normally use `gtk_range_set_value()`. To detect changes to the value, you would normally use the "value-changed" signal.

Note that using the same upper and lower bounds for the GtkScale (through the GtkRange methods) will hide the slider itself. This is useful for applications that want to show an undeterminate value on the scale, without changing the layout of the application (such as movie or music players).

GtkScale as GtkBuildable

GtkScale supports a custom <marks> element, which can contain multiple <mark> elements. The "value" and "position" attributes have the same meaning as `gtk_scale_add_mark()` parameters of the same name. If the element is not empty, its content is taken as the markup to show at the mark. It can be translated with the usual "translatable" and "context" attributes.

CSS nodes

```
scale[.fine-tune][.marks-before][.marks-after]
├── marks.top
│   ├── mark
│   │   ├── [label]
│   │   └── indicator
┆   ┆
│   └── mark
├── [value]
├── contents
│   └── trough
│       ├── slider
│       ├── [highlight]
│       └── [fill]
└── marks.bottom
    ├── mark
    │   ├── indicator
    │   └── [label]
    └── mark
```

GtkScale has a main CSS node with name scale and a subnode for its contents, with subnodes named trough and slider.

The main node gets the style class .fine-tune added when the scale is in 'fine-tuning' mode.

If the scale has an origin (see `gtk_scale_set_has_origin()`), there is a subnode with name highlight below the trough node that is used for rendering the highlighted part of the trough.

If the scale is showing a fill level (see `gtk_range_set_show_fill_level()`), there is a subnode with name fill below the trough node that is used for rendering the filled in part of the trough.

If marks are present, there is a marks subnode before or after the contents node, below which each mark gets a node with name mark. The marks nodes get either the .top or .bottom style class.

The mark node has a subnode named indicator. If the mark has text, it also has a subnode named label. When the mark is either above or left of the scale, the label subnode is the first when present. Otherwise, the indicator subnode is the first.

The main CSS node gets the 'marks-before' and/or 'marks-after' style classes added depending on what marks are present.

If the scale is displaying the value (see “draw-value”), there is subnode with name value.

Functions

gtk_scale_new ()

```
GtkWidget *
gtk_scale_new (GtkOrientation orientation,
               GtkAdjustment *adjustment);
```

Creates a new GtkScale.

Parameters

orientation	the scale's orientation.	
adjustment	the GtkAdjustment which sets the range of the scale, or `NULL` to create a new adjustment.	[nullable]

Returns

a new GtkScale

Since: 3.0

---

gtk_scale_new_with_range ()

```
GtkWidget *
gtk_scale_new_with_range (GtkOrientation orientation,
                          gdouble min,
                          gdouble max,
                          gdouble step);
```

Creates a new scale widget with the given orientation that lets the user input a number between `min` and `max` (including `min` and `max`) with the increment `step` . `step` must be nonzero; it's the distance the slider moves when using the arrow keys to adjust the scale value.

Note that the way in which the precision is derived works best if `step` is a power of ten. If the resulting precision is not suitable for your needs, use `gtk_scale_set_digits()` to correct it.

Parameters

orientation	the scale's orientation.
min	minimum value
max	maximum value
step	step increment (tick size) used with keyboard shortcuts

Returns

a new GtkScale

Since: 3.0

---

gtk_scale_set_digits ()

```
void
gtk_scale_set_digits (GtkScale *scale,
                      gint digits);
```

Sets the number of decimal places that are displayed in the value. Also causes the value of the adjustment to be rounded to this number of digits, so the retrieved value matches the displayed one, if "draw-value" is `TRUE` when the value changes. If you want to enforce rounding the value when "draw-value" is `FALSE`, you can set

“round-digits” instead.

Note that rounding to a small number of digits can interfere with the smooth autoscrolling that is built into GtkScale. As an alternative, you can use the “format-value” signal to format the displayed value yourself.

Parameters

scale	a GtkScale
digits	the number of decimal places to display, e.g. use 1 to display 1.0, 2 to display 1.00, etc

---

gtk_scale_set_draw_value ()

```
void
gtk_scale_set_draw_value (GtkScale *scale,
                          gboolean draw_value);
```

Specifies whether the current value is displayed as a string next to the slider.

Parameters

scale	a GtkScale
draw_value	TRUE to draw the value

---

gtk_scale_set_has_origin ()

```
void
gtk_scale_set_has_origin (GtkScale *scale,
                          gboolean has_origin);
```

If “has-origin” is set to TRUE (the default), the scale will highlight the part of the trough between the origin (bottom or left side) and the current value.

Parameters

scale	a GtkScale
has_origin	TRUE if the scale has an origin

Since: 3.4

---

gtk_scale_set_value_pos ()

```
void
gtk_scale_set_value_pos (GtkScale *scale,
                         GtkPositionType pos);
```

Sets the position in which the current value is displayed.

Parameters

scale	a GtkScale
pos	the position in which the current value is displayed

---

gtk_scale_get_digits ()

```
gint
gtk_scale_get_digits (GtkScale *scale);
```

Gets the number of decimal places that are displayed in the value.

Parameters

scale	a GtkScale

Returns

the number of decimal places that are displayed

---

gtk_scale_get_draw_value ()

```
gboolean
gtk_scale_get_draw_value (GtkScale *scale);
```

Returns whether the current value is displayed as a string next to the slider.

Parameters

scale	a GtkScale

Returns

whether the current value is displayed as a string

---

gtk_scale_get_has_origin ()

```
gboolean
gtk_scale_get_has_origin (GtkScale *scale);
```

Returns whether the scale has an origin.

Parameters

scale	a GtkScale

Returns

TRUE if the scale has an origin.

Since: 3.4

---

gtk_scale_get_value_pos ()

```
GtkPositionType
gtk_scale_get_value_pos (GtkScale *scale);
```

Gets the position in which the current value is displayed.

Parameters

scale	a GtkScale

Returns

the position in which the current value is displayed

---

gtk_scale_get_layout ()

```
PangoLayout *
gtk_scale_get_layout (GtkScale *scale);
```

Gets the PangoLayout used to display the scale. The returned object is owned by the scale so does not need to be freed by the caller.

Parameters

scale	A GtkScale

Returns

the PangoLayout for this scale, or NULL if the “draw-value” property is FALSE.

[transfer none][nullable]

Since: 2.4

---

gtk_scale_get_layout_offsets ()

```
void
gtk_scale_get_layout_offsets (GtkScale *scale,
                              gint *x,
                              gint *y);
```

Obtains the coordinates where the scale will draw the PangoLayout representing the text in the scale. Remember when using the PangoLayout function you need to convert to and from pixels using `PANGO_PIXELS()` or PANGO_SCALE.

If the "draw-value" property is `FALSE`, the return values are undefined.

Parameters

scale	a GtkScale	
x	location to store X offset of layout, or `NULL`.	[out][allow-none]
y	location to store Y offset of layout, or `NULL`.	[out][allow-none]

Since: 2.4

---

gtk_scale_add_mark ()

```
void
gtk_scale_add_mark (GtkScale *scale,
                    gdouble value,
                    GtkPositionType position,
                    const gchar *markup);
```

Adds a mark at `value` .

A mark is indicated visually by drawing a tick mark next to the scale, and GTK+ makes it easy for the user to position the scale exactly at the marks value.

If `markup` is not `NULL`, text is shown next to the tick mark.

To remove marks from a scale, use `gtk_scale_clear_marks()`.

Parameters

scale	a GtkScale
value	the value at which the mark is placed, must be between the lower and upper limits of the scales' adjustment
position	where to draw the mark. For a horizontal scale, GTK_POS_TOP and `GTK_POS_LEFT` are drawn above the scale, anything else below. For a vertical scale, GTK_POS_LEFT and `GTK_POS_TOP` are drawn to the left of the scale, anything else to the

	right.	
markup	Text to be shown at the mark, using Pango markup, or `NULL`.	[allow-none]

Since: 2.16

---

gtk_scale_clear_marks ()

```
void
gtk_scale_clear_marks (GtkScale *scale);
```

Removes any marks that have been added with `gtk_scale_add_mark()`.

Parameters

scale	a GtkScale	

Since: 2.16

Types and Values

struct GtkScale

```
struct GtkScale;
```

Property Details

The “digits” property

```
  “digits”                   gint
```

The number of decimal places that are displayed in the value.

Flags: Read / Write

Allowed values: [-1,64]

Default value: 1

The “draw-value” property

```
  “draw-value”               gboolean
```

Whether the current value is displayed as a string next to the slider.

Flags: Read / Write

Default value: TRUE

The “has-origin” property

“has-origin” gboolean

Whether the scale has an origin.

Flags: Read / Write

Default value: TRUE

The “value-pos” property

“value-pos” GtkPositionType

The position in which the current value is displayed.

Flags: Read / Write

Default value: GTK_POS_TOP

Style Property Details

The “slider-length” style property

“slider-length” gint

Length of scale's slider.

GtkScale:slider-length has been deprecated since version 3.20 and should not be used in newly-written code.

Use min-height/min-width CSS properties on the slider element instead. The value of this style property is ignored.

Flags: Read

Allowed values: >= 0

Default value: 31

The “value-spacing” style property

“value-spacing” gint

Space between value text and the slider/trough area.

GtkScale:value-spacing has been deprecated since version 3.20 and should not be used in newly-written code.

Use min-height/min-width CSS properties on the value element instead. The value of this style property is ignored.

Flags: Read

Allowed values: >= 0

Default value: 2

Signal Details

The `"format-value"` signal

```
gchar*
user_function (GtkScale *scale,
               gdouble   value,
               gpointer  user_data)
```

Signal which allows you to change how the scale value is displayed. Connect a signal handler which returns an allocated string representing `value` . That string will then be used to display the scale's value.

If no user-provided handlers are installed, the value will be displayed on its own, rounded according to the value of the "digits" property.

Here's an example signal handler which displays a value 1.0 as with "-->1.0<--".

```
static gchar*
format_value_callback (GtkScale *scale,
                       gdouble   value)
{
  return g_strdup_printf ("-->\%0.*g<--",

gtk_scale_get_digits (scale), value);
 }
```

Parameters

scale	the object which received the signal
value	the value to format
user_data	user data set when the signal handler was connected.

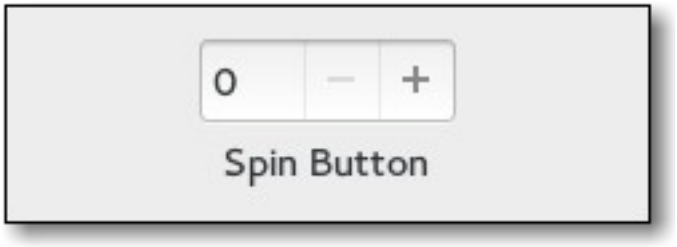
Returns

allocated string representing `value`

Flags: Run Last

GtkSpinButton

GtkSpinButton — Retrieve an integer or floating-point number from the user



Functions

void	gtk_spin_button_configure ()
GtkWidget *	gtk_spin_button_new ()
GtkWidget *	gtk_spin_button_new_with_range ()
void	gtk_spin_button_set_adjustment ()
GtkAdjustment *	gtk_spin_button_get_adjustment ()
void	gtk_spin_button_set_digits ()
void	gtk_spin_button_set_increments ()
void	gtk_spin_button_set_range ()
gint	gtk_spin_button_get_value_as_int ()
void	gtk_spin_button_set_value ()
void	gtk_spin_button_set_update_policy ()
void	gtk_spin_button_set_numeric ()
void	gtk_spin_button_spin ()
void	gtk_spin_button_set_wrap ()
void	gtk_spin_button_set_snap_to_ticks ()
void	gtk_spin_button_update ()
guint	gtk_spin_button_get_digits ()
void	gtk_spin_button_get_increments ()
gboolean	gtk_spin_button_get_numeric ()
void	gtk_spin_button_get_range ()
gboolean	gtk_spin_button_get_snap_to_ticks ()
GtkSpinButtonUpdatePolicy	gtk_spin_button_get_update_policy ()
gdouble	gtk_spin_button_get_value ()
gboolean	gtk_spin_button_get_wrap ()

Properties

GtkAdjustment *	adjustment	Read / Write
gdouble	climb-rate	Read / Write
guint	digits	Read / Write
gboolean	numeric	Read / Write
gboolean	snap-to-ticks	Read / Write
GtkSpinButtonUpdatePolicy	update-policy	Read / Write
gdouble	value	Read / Write
gboolean	wrap	Read / Write

Style Properties

GtkShadowType	shadow-type	Read

Signals

void	change-value	Action
gint	input	Run Last
gboolean	output	Run Last
void	value-changed	Run Last
void	wrapped	Run Last

Types and Values

struct	GtkSpinButton
enum	GtkSpinButtonUpdatePolicy
enum	GtkSpinType
#define	GTK_INPUT_ERROR

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkEntry
            └── GtkSpinButton
```

Implemented Interfaces

GtkSpinButton implements AtkImplementorIface, GtkBuildable, GtkEditable, GtkCellEditable and GtkOrientable.

Includes

```
#include <gtk/gtk.h>
```

Description

A GtkSpinButton is an ideal way to allow the user to set the value of some attribute. Rather than having to directly type a number into a GtkEntry, GtkSpinButton allows the user to click on one of two arrows to increment or decrement the displayed value. A value can still be typed in, with the bonus that it can be checked to ensure it is in a given range.

The main properties of a GtkSpinButton are through an adjustment. See the GtkAdjustment section for more details about an adjustment's properties. Note that GtkSpinButton will by default make its entry large enough to accomodate the lower and upper bounds of the adjustment, which can lead to surprising results. Best practice is to set both the “width-chars” and “max-width-chars” poperties to the desired number of characters to display in the entry.

CSS nodes

```
spinbutton.horizontal
├── undershoot.left
├── undershoot.right
├── entry
│   └── ...
├── button.down
└── button.up
spinbutton.vertical
├── undershoot.left
├── undershoot.right
├── button.up
├── entry
│   └── ...
└── button.down
```

GtkSpinButtons main CSS node has the name spinbutton. It creates subnodes for the entry and the two buttons, with these names. The button nodes have the style classes .up and .down. The GtkEntry subnodes (if present) are put below the entry node. The orientation of the spin button is reflected in the .vertical or .horizontal style class on the main node.

Using a GtkSpinButton to get an integer

```
// Provides a function to retrieve an integer
value from a GtkSpinButton
// and creates a spin button to model
percentage values.

gint
grab_int_value (GtkSpinButton *button,
                gpointer       user_data)
{
  return gtk_spin_button_get_value_as_int
(button);
}

void
create_integer_spin_button (void)
{

  GtkWidget *window, *button;
  GtkAdjustment *adjustment;

  adjustment = gtk_adjustment_new (50.0, 0.0,
100.0, 1.0, 5.0, 0.0);

  window = gtk_window_new
(GTK_WINDOW_TOPLEVEL);
  gtk_container_set_border_width
(GTK_CONTAINER (window), 5);

  // creates the spinbutton, with no decimal
places
  button = gtk_spin_button_new (adjustment,
1.0, 0);
  gtk_container_add (GTK_CONTAINER (window),
button);

  gtk_widget_show_all (window);
}
```

Using a GtkSpinButton to get a floating point value

```
1                                                   // Provides a function to retrieve a floating
2                                                   point value from a
3                                                   // GtkSpinButton, and creates a high
4                                                   precision spin button.
5
6                                                   gfloat
7                                                   grab_float_value (GtkSpinButton *button,
8                                                                     gpointer       user_data)
9                                                   {
10                                                    return gtk_spin_button_get_value (button);
11                                                  }
12
13                                                  void
14                                                  create_floating_spin_button (void)
15                                                  {
16                                                    GtkWidget *window, *button;
17                                                    GtkAdjustment *adjustment;
18
19                                                    adjustment = gtk_adjustment_new (2.500,
20                                                  0.0, 5.0, 0.001, 0.1, 0.0);
21
22                                                    window = gtk_window_new
23                                                  (GTK_WINDOW_TOPLEVEL);
24                                                    gtk_container_set_border_width
25                                                  (GTK_CONTAINER (window), 5);
26
27                                                    // creates the spinbutton, with three
                                                    decimal places
                                                    button = gtk_spin_button_new (adjustment,
                                                    0.001, 3);
                                                    gtk_container_add (GTK_CONTAINER (window),
                                                    button);

                                                    gtk_widget_show_all (window);
                                                  }
```

Functions

gtk_spin_button_configure ()

```
void
gtk_spin_button_configure (GtkSpinButton *spin_button,
                           GtkAdjustment *adjustment,
                           gdouble climb_rate,
                           guint digits);
```

Changes the properties of an existing spin button. The adjustment, climb rate, and number of decimal places are updated accordingly.

Parameters

spin_button	a GtkSpinButton	
adjustment	a GtkAdjustment to replace the spin button's existing adjustment, or	[nullable]

	NULL to leave its current adjustment unchanged.
climb_rate	the new climb rate
digits	the number of decimal places to display in the spin button

---

gtk_spin_button_new ()

```
GtkWidget *
gtk_spin_button_new (GtkAdjustment *adjustment,
                     gdouble climb_rate,
                     guint digits);
```

Creates a new GtkSpinButton.

Parameters

adjustment	the GtkAdjustment object that this spin button should use, or NULL.	[allow-none]
climb_rate	specifies by how much the rate of change in the value will accelerate if you continue to hold down an up/down button or arrow key	
digits	the number of decimal places to display	

Returns

The new spin button as a GtkWidget

---

gtk_spin_button_new_with_range ()

```
GtkWidget *
gtk_spin_button_new_with_range (gdouble min,
                                gdouble max,
                                gdouble step);
```

This is a convenience constructor that allows creation of a numeric GtkSpinButton without manually creating an adjustment. The value is initially set to the minimum value and a page increment of 10 * `step` is the default. The precision of the spin button is equivalent to the precision of `step` .

Note that the way in which the precision is derived works best if `step` is a power of ten. If the resulting precision is not suitable for your needs, use `gtk_spin_button_set_digits()` to correct it.

Parameters

min	Minimum allowable value
max	Maximum allowable value
step	Increment added or subtracted by

spinning the widget

Returns

The new spin button as a GtkWidget

---

gtk_spin_button_set_adjustment ()

```
void
gtk_spin_button_set_adjustment (GtkSpinButton *spin_button,
                                GtkAdjustment *adjustment);
```

Replaces the GtkAdjustment associated with `spin_button` .

Parameters

spin_button	a GtkSpinButton
adjustment	a GtkAdjustment to replace the existing adjustment

---

gtk_spin_button_get_adjustment ()

```
GtkAdjustment *
gtk_spin_button_get_adjustment (GtkSpinButton *spin_button);
```

Get the adjustment associated with a GtkSpinButton

Parameters

spin_button	a GtkSpinButton

Returns

the GtkAdjustment of `spin_button` .

[transfer none]

---

gtk_spin_button_set_digits ()

```
void
gtk_spin_button_set_digits (GtkSpinButton *spin_button,
                            guint digits);
```

Set the precision to be displayed by `spin_button` . Up to 20 digit precision is allowed.

Parameters

spin_button	a GtkSpinButton
digits	the number of digits after the decimal point to be displayed for the spin button's value

---

gtk_spin_button_set_increments ()

```
void
gtk_spin_button_set_increments (GtkSpinButton *spin_button,
                                gdouble step,
                                gdouble page);
```

Sets the step and page increments for spin_button. This affects how quickly the value changes when the spin button's arrows are activated.

Parameters

spin_button	a GtkSpinButton
step	increment applied for a button 1 press.
page	increment applied for a button 2 press.

---

gtk_spin_button_set_range ()

```
void
gtk_spin_button_set_range (GtkSpinButton *spin_button,
                           gdouble min,
                           gdouble max);
```

Sets the minimum and maximum allowable values for `spin_button` .

If the current value is outside this range, it will be adjusted to fit within the range, otherwise it will remain unchanged.

Parameters

spin_button	a GtkSpinButton
min	minimum allowable value
max	maximum allowable value

---

gtk_spin_button_get_value_as_int ()

```
gint
gtk_spin_button_get_value_as_int (GtkSpinButton *spin_button);
```

Get the value `spin_button` represented as an integer.

Parameters

spin_button	a GtkSpinButton

Returns

the value of `spin_button`

---

gtk_spin_button_set_value ()

```
void
gtk_spin_button_set_value (GtkSpinButton *spin_button,
                           gdouble value);
```

Sets the value of `spin_button` .

Parameters

spin_button	a GtkSpinButton
value	the new value

---

gtk_spin_button_set_update_policy ()

```
void
gtk_spin_button_set_update_policy (GtkSpinButton *spin_button,
                                   GtkSpinButtonUpdatePolicy policy);
```

Sets the update behavior of a spin button. This determines whether the spin button is always updated or only when a valid value is set.

Parameters

spin_button	a GtkSpinButton
policy	a GtkSpinButtonUpdatePolicy value

---

gtk_spin_button_set_numeric ()

```
void
gtk_spin_button_set_numeric (GtkSpinButton *spin_button,
                             gboolean numeric);
```

Sets the flag that determines if non-numeric text can be typed into the spin button.

Parameters

spin_button	a GtkSpinButton
numeric	flag indicating if only numeric entry is allowed

---

gtk_spin_button_spin ()

```
void
gtk_spin_button_spin (GtkSpinButton *spin_button,
                      GtkSpinType direction,
                      gdouble increment);
```

Increment or decrement a spin button's value in a specified direction by a specified amount.

Parameters

spin_button	a GtkSpinButton
direction	a GtkSpinType indicating the direction to spin
increment	step increment to apply in the specified direction

---

gtk_spin_button_set_wrap ()

```
void
gtk_spin_button_set_wrap (GtkSpinButton *spin_button,
                          gboolean wrap);
```

Sets the flag that determines if a spin button value wraps around to the opposite limit when the upper or lower limit of the range is exceeded.

Parameters

spin_button	a GtkSpinButton
wrap	a flag indicating if wrapping behavior is performed

---

gtk_spin_button_set_snap_to_ticks ()

```
void
gtk_spin_button_set_snap_to_ticks (GtkSpinButton *spin_button,
                                   gboolean snap_to_ticks);
```

Sets the policy as to whether values are corrected to the nearest step increment when a spin button is activated after providing an invalid value.

Parameters

spin_button	a GtkSpinButton	
snap_to_ticks	a flag indicating if invalid values should be corrected	

---

gtk_spin_button_update ()

```
void
gtk_spin_button_update (GtkSpinButton *spin_button);
```

Manually force an update of the spin button.

Parameters

spin_button	a GtkSpinButton	

---

gtk_spin_button_get_digits ()

```
guint
gtk_spin_button_get_digits (GtkSpinButton *spin_button);
```

Fetches the precision of `spin_button` . See `gtk_spin_button_set_digits()`.

Parameters

spin_button	a GtkSpinButton	

Returns

the current precision

---

gtk_spin_button_get_increments ()

```
void
gtk_spin_button_get_increments (GtkSpinButton *spin_button,
                                gdouble *step,
                                gdouble *page);
```

Gets the current step and page the increments used by `spin_button` . See `gtk_spin_button_set_increments()`.

Parameters

spin_button	a GtkSpinButton	
step	location to store step increment, or `NULL`.	[out][allow-none]

page	location to store page increment, or `NULL`.	[out][allow-none]

---

gtk_spin_button_get_numeric ()

```
gboolean
gtk_spin_button_get_numeric (GtkSpinButton *spin_button);
```

Returns whether non-numeric text can be typed into the spin button. See `gtk_spin_button_set_numeric()`.

Parameters

spin_button	a GtkSpinButton	

Returns

`TRUE` if only numeric text can be entered

---

gtk_spin_button_get_range ()

```
void
gtk_spin_button_get_range (GtkSpinButton *spin_button,
                           gdouble *min,
                           gdouble *max);
```

Gets the range allowed for `spin_button` . See `gtk_spin_button_set_range()`.

Parameters

spin_button	a GtkSpinButton	
min	location to store minimum allowed value, or `NULL`.	[out][allow-none]
max	location to store maximum allowed value, or `NULL`.	[out][allow-none]

---

gtk_spin_button_get_snap_to_ticks ()

```
gboolean
gtk_spin_button_get_snap_to_ticks (GtkSpinButton *spin_button);
```

Returns whether the values are corrected to the nearest step. See `gtk_spin_button_set_snap_to_ticks()`.

Parameters

spin_button	a GtkSpinButton	

Returns

`TRUE` if values are snapped to the nearest step

---

gtk_spin_button_get_update_policy ()

```
GtkSpinButtonUpdatePolicy
gtk_spin_button_get_update_policy (GtkSpinButton *spin_button);
```

Gets the update behavior of a spin button. See `gtk_spin_button_set_update_policy()`.

Parameters

spin_button	a GtkSpinButton

Returns

the current update policy

---

gtk_spin_button_get_value ()

```
gdouble
gtk_spin_button_get_value (GtkSpinButton *spin_button);
```

Get the value in the `spin_button` .

Parameters

spin_button	a GtkSpinButton

Returns

the value of `spin_button`

---

gtk_spin_button_get_wrap ()

```
gboolean
gtk_spin_button_get_wrap (GtkSpinButton *spin_button);
```

Returns whether the spin button's value wraps around to the opposite limit when the upper or lower limit of the range is exceeded. See `gtk_spin_button_set_wrap()`.

Parameters

spin_button	a GtkSpinButton

Returns

`TRUE` if the spin button wraps around

Types and Values

struct GtkSpinButton

```
struct GtkSpinButton;
```

The GtkSpinButton contains only private data and should not be directly modified.

---

enum GtkSpinButtonUpdatePolicy

The spin button update policy determines whether the spin button displays values even if they are outside the bounds of its adjustment. See `gtk_spin_button_set_update_policy()`.

Members

GTK_UPDATE_ALWAYS	When refreshing your GtkSpinButton, the value is always displayed
GTK_UPDATE_IF_VALID	When refreshing your GtkSpinButton, the value is only displayed if it is valid within the bounds of the spin button's adjustment

---

enum GtkSpinType

The values of the GtkSpinType enumeration are used to specify the change to make in `gtk_spin_button_spin()`.

Members

GTK_SPIN_STEP_FORWARD	Increment by the adjustments step increment.
GTK_SPIN_STEP_BACKWARD	Decrement by the adjustments step increment.
GTK_SPIN_PAGE_FORWARD	Increment by the adjustments page increment.
GTK_SPIN_PAGE_BACKWARD	Decrement by the adjustments page increment.
GTK_SPIN_HOME	Go to the adjustments lower bound.
GTK_SPIN_END	Go to the adjustments upper bound.

GTK_SPIN_USER_DEFINED	Change by a specified amount.

GTK_INPUT_ERROR

```
#define GTK_INPUT_ERROR -1
```

Constant to return from a signal handler for the “input” signal in case of conversion failure.

Property Details

The “adjustment” property

```
  “adjustment”               GtkAdjustment *
```

The adjustment that holds the value of the spin button.

Flags: Read / Write

The “climb-rate” property

```
  “climb-rate”               gdouble
```

The acceleration rate when you hold down a button or key.

Flags: Read / Write

Allowed values: >= 0

Default value: 0

The “digits” property

```
  “digits”                   guint
```

The number of decimal places to display.

Flags: Read / Write

Allowed values: <= 20

Default value: 0

The “numeric” property

```
  “numeric”                  gboolean
```

Whether non-numeric characters should be ignored.

Flags: Read / Write

Default value: FALSE

The “snap-to-ticks” property

“snap-to-ticks” gboolean

Whether erroneous values are automatically changed to a spin button's nearest step increment.

Flags: Read / Write

Default value: FALSE

The “update-policy” property

“update-policy” GtkSpinButtonUpdatePolicy

Whether the spin button should update always, or only when the value is legal.

Flags: Read / Write

Default value: GTK_UPDATE_ALWAYS

The “value” property

“value” gdouble

Reads the current value, or sets a new value.

Flags: Read / Write

Default value: 0

The “wrap” property

“wrap” gboolean

Whether a spin button should wrap upon reaching its limits.

Flags: Read / Write

Default value: FALSE

Style Property Details

The “shadow-type” style property

“shadow-type” GtkShadowType

Style of bevel around the spin button.

GtkSpinButton:shadow-type has been deprecated since version 3.20 and should not be used in newly-written code.

Use CSS to determine the style of the border; the value of this style property is ignored.

Flags: Read

Default value: GTK_SHADOW_IN

Signal Details

The "change-value" signal

```
void
user_function (GtkSpinButton *spin_button,
               GtkScrollType  scroll,
               gpointer       user_data)
```

The ::change-value signal is a keybinding signal which gets emitted when the user initiates a value change.

Applications should not connect to it, but may emit it with g_signal_emit_by_name() if they need to control the cursor programmatically.

The default bindings for this signal are Up/Down and PageUp and/PageDown.

Parameters

spin_button	the object on which the signal was emitted
scroll	a GtkScrollType to specify the speed and amount of change
user_data	user data set when the signal handler was connected.

Flags: Action

The "input" signal

```
gint
user_function (GtkSpinButton *spin_button,
               gpointer       new_value,
               gpointer       user_data)
```

The ::input signal can be used to influence the conversion of the users input into a double value. The signal handler is expected to use gtk_entry_get_text() to retrieve the text of the entry and set new_value to the new value.

The default conversion uses g_strtod().

Parameters

spin_button	the object on which the signal was emitted	
new_value	return location for the new value.	[out][type double]
user_data	user data set when the signal handler was connected.	

Returns

`TRUE` for a successful conversion, `FALSE` if the input was not handled, and `GTK_INPUT_ERROR` if the conversion failed.

Flags: Run Last

The "output" signal

```
gboolean
user_function (GtkSpinButton *spin_button,
               gpointer       user_data)
```

The ::output signal can be used to change to formatting of the value that is displayed in the spin buttons entry.

```
// show leading zeros
static gboolean
on_output (GtkSpinButton *spin,
           gpointer       data)
{
   GtkAdjustment *adjustment;
   gchar *text;
   int value;

   adjustment =
gtk_spin_button_get_adjustment (spin);
   value = (int)gtk_adjustment_get_value
(adjustment);
   text = g_strdup_printf ("%02d", value);
   gtk_entry_set_text (GTK_ENTRY (spin),
text);
   g_free (text);

   return TRUE;
}
```

Parameters

spin_button	the object on which the signal was emitted
user_data	user data set when the signal handler was connected.

Returns

`TRUE` if the value has been displayed

Flags: Run Last

The “value-changed” signal

```
void
user_function (GtkSpinButton *spin_button,
               gpointer       user_data)
```

The ::value-changed signal is emitted when the value represented by `spinbutton` changes. Also see the “output” signal.

Parameters

spin_button	the object on which the signal was emitted
user_data	user data set when the signal handler was connected.

Flags: Run Last

The “wrapped” signal

```
void
user_function (GtkSpinButton *spin_button,
               gpointer       user_data)
```

The ::wrapped signal is emitted right after the spinbutton wraps from its maximum to minimum value or vice-versa.

Parameters

spin_button	the object on which the signal was emitted
user_data	user data set when the signal handler was connected.

Flags: Run Last

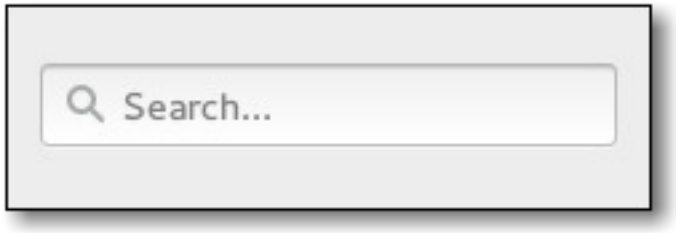
Since: 2.10

See Also

GtkEntry

GtkSearchEntry

GtkSearchEntry — An entry which shows a search icon



Functions

GtkWidget *	gtk_search_entry_new ()
gboolean	gtk_search_entry_handle_event ()

Signals

void	next-match	Action
void	previous-match	Action
void	search-changed	Run Last
void	stop-search	Action

Types and Values

struct	GtkSearchEntry

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkEntry
            └── GtkSearchEntry
```

Implemented Interfaces

GtkSearchEntry implements AtkImplementorIface, GtkBuildable, GtkEditable and GtkCellEditable.

Includes

```
#include <gtk/gtk.h>
```

Description

GtkSearchEntry is a subclass of GtkEntry that has been tailored for use as a search entry.

It will show an inactive symbolic “find” icon when the search entry is empty, and a symbolic “clear” icon when there is text. Clicking on the “clear” icon will empty the search entry.

Note that the search/clear icon is shown using a secondary icon, and thus does not work if you are using the secondary icon position for some other purpose.

To make filtering appear more reactive, it is a good idea to not react to every change in the entry text immediately, but only after a short delay. To support this, GtkSearchEntry emits the “search-changed” signal which can be used instead of the “changed” signal.

The “previous-match”, “next-match” and “stop-search” signals can be used to implement moving between search results and ending the search.

Often, GtkSearchEntry will be fed events by means of being placed inside a GtkSearchBar. If that is not the case, you can use `gtk_search_entry_handle_event()` to pass events.

Functions

gtk_search_entry_new ()

```
GtkWidget *
gtk_search_entry_new (void);
```

Creates a GtkSearchEntry, with a find icon when the search field is empty, and a clear icon when it isn't.

Returns

a new GtkSearchEntry

Since: 3.6

---

gtk_search_entry_handle_event ()

```
gboolean
gtk_search_entry_handle_event (GtkSearchEntry *entry,
                               GdkEvent *event);
```

This function should be called when the top-level window which contains the search entry received a key event. If the entry is part of a GtkSearchBar, it is preferable to call `gtk_search_bar_handle_event()` instead, which will reveal the entry in addition to passing the event to this function.

If the key event is handled by the search entry and starts or continues a search, `GDK_EVENT_STOP` will be returned. The caller should ensure that the entry is shown in this case, and not propagate the event further.

Parameters

entry	a GtkSearchEntry
event	a key event

Returns

`GDK_EVENT_STOP` if the key press event resulted in a search beginning or continuing, `GDK_EVENT_PROPAGATE`

otherwise.

Since: 3.16

Types and Values

struct GtkSearchEntry

```
struct GtkSearchEntry;
```

Signal Details

The “next-match” signal

```
void
user_function (GtkSearchEntry *entry,
               gpointer        user_data)
```

The ::next-match signal is a keybinding signal which gets emitted when the user initiates a move to the next match for the current search string.

Applications should connect to it, to implement moving between matches.

The default bindings for this signal is Ctrl-g.

Parameters

entry	the entry on which the signal was emitted
user_data	user data set when the signal handler was connected.

Flags: Action

Since: 3.16

The “previous-match” signal

```
void
user_function (GtkSearchEntry *entry,
               gpointer        user_data)
```

The ::previous-match signal is a keybinding signal which gets emitted when the user initiates a move to the previous match for the current search string.

Applications should connect to it, to implement moving between matches.

The default bindings for this signal is Ctrl-Shift-g.

Parameters

entry	the entry on which the signal was emitted
user_data	user data set when the signal handler was connected.

Flags: Action

Since: 3.16

The `"search-changed"` signal

```
void
user_function (GtkSearchEntry *entry,
               gpointer        user_data)
```

The "search-changed" signal is emitted with a short delay of 150 milliseconds after the last change to the entry text.

Parameters

entry	the entry on which the signal was emitted
user_data	user data set when the signal handler was connected.

Flags: Run Last

Since: 3.10

The `"stop-search"` signal

```
void
user_function (GtkSearchEntry *entry,
               gpointer        user_data)
```

The ::stop-search signal is a keybinding signal which gets emitted when the user stops a search via keyboard input.

Applications should connect to it, to implement hiding the search entry in this case.

The default bindings for this signal is Escape.

Parameters

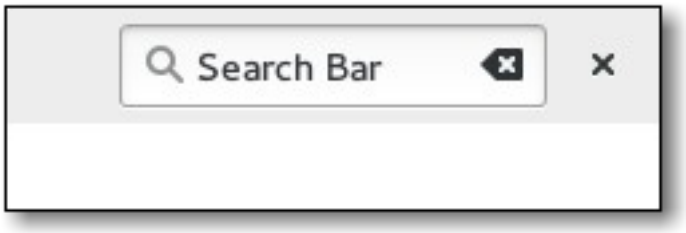
entry	the entry on which the signal was emitted
user_data	user data set when the signal handler was connected.

Flags: Action

Since: 3.16

GtkSearchBar

GtkSearchBar — A toolbar to integrate a search entry with



Functions

GtkWidget *	gtk_search_bar_new ()
void	gtk_search_bar_connect_entry ()
gboolean	gtk_search_bar_get_search_mode ()
void	gtk_search_bar_set_search_mode ()
gboolean	gtk_search_bar_get_show_close_button ()
void	gtk_search_bar_set_show_close_button ()
gboolean	gtk_search_bar_handle_event ()

Properties

gboolean	search-mode-enabled	Read / Write
gboolean	show-close-button	Read / Write / Construct

Types and Values

struct	GtkSearchBar
struct	GtkSearchBarClass

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkContainer
            └── GtkBin
                └── GtkSearchBar
```

Implemented Interfaces

GtkSearchBar implements AtkImplementorIface and GtkBuildable.

Includes

```
#include <gtk/gtk.h>
```

Description

GtkSearchBar is a container made to have a search entry (possibly with additional connex widgets, such as drop-down menus, or buttons) built-in. The search bar would appear when a search is started through typing on the keyboard, or the application's search mode is toggled on.

For keyboard presses to start a search, events will need to be forwarded from the top-level window that contains the search bar. See `gtk_search_bar_handle_event()` for example code. Common shortcuts such as Ctrl+F should be handled as an application action, or through the menu items.

You will also need to tell the search bar about which entry you are using as your search entry using `gtk_search_bar_connect_entry()`. The following example shows you how to create a more complex search entry.

CSS nodes

GtkSearchBar has a single CSS node with name searchbar.

Creating a search bar

A simple example

Functions

gtk_search_bar_new ()

```
GtkWidget *
gtk_search_bar_new (void);
```

Creates a GtkSearchBar. You will need to tell it about which widget is going to be your text entry using `gtk_search_bar_connect_entry()`.

Returns

a new GtkSearchBar

Since: 3.10

---

gtk_search_bar_connect_entry ()

```
void
gtk_search_bar_connect_entry (GtkSearchBar *bar,
                              GtkEntry *entry);
```

Connects the GtkEntry widget passed as the one to be used in this search bar. The entry should be a descendant of the search bar. This is only required if the entry isn't the direct child of the search bar (as in our main example).

Parameters

bar	a GtkSearchBar
entry	a GtkEntry

Since: 3.10

---

gtk_search_bar_get_search_mode ()

```
gboolean
gtk_search_bar_get_search_mode (GtkSearchBar *bar);
```

Returns whether the search mode is on or off.

Parameters

bar	a GtkSearchBar

Returns

whether search mode is toggled on

Since: 3.10

---

gtk_search_bar_set_search_mode ()

```
void
gtk_search_bar_set_search_mode (GtkSearchBar *bar,
                                gboolean search_mode);
```

Switches the search mode on or off.

Parameters

bar	a GtkSearchBar
search_mode	the new state of the search mode

Since: 3.10

---

gtk_search_bar_get_show_close_button ()

```
gboolean
gtk_search_bar_get_show_close_button (GtkSearchBar *bar);
```

Returns whether the close button is shown.

Parameters

bar	a GtkSearchBar

Returns

whether the close button is shown

Since: 3.10

---

gtk_search_bar_set_show_close_button ()

```
void
gtk_search_bar_set_show_close_button (GtkSearchBar *bar,
                                      gboolean visible);
```

Shows or hides the close button. Applications that already have a “search” toggle button should not show a close button in their search bar, as it duplicates the role of the toggle button.

Parameters

bar	a GtkSearchBar
visible	whether the close button will be shown or not

Since: 3.10

---

gtk_search_bar_handle_event ()

```
gboolean
gtk_search_bar_handle_event (GtkSearchBar *bar,
                             GdkEvent *event);
```

This function should be called when the top-level window which contains the search bar received a key event.

If the key event is handled by the search bar, the bar will be shown, the entry populated with the entered text and `GDK_EVENT_STOP` will be returned. The caller should ensure that events are not propagated further.

If no entry has been connected to the search bar, using `gtk_search_bar_connect_entry()`, this function will return immediately with a warning.

Showing the search bar on key presses

```
1    static gboolean
2    on_key_press_event (GtkWidget *widget,
3                        GdkEvent  *event,
4                        gpointer   user_data)
5    {
6      GtkSearchBar *bar = GTK_SEARCH_BAR (user_data);
7      return gtk_search_bar_handle_event (bar, event);
8    }
9
10   static void
11   create_toplevel (void)
12   {
13     GtkWidget *window = gtk_window_new (GTK_WINDOW_TOPLEVEL);
14     GtkWindow *search_bar = gtk_search_bar_new ();
15
16    // Add more widgets to the window...
17
18     g_signal_connect (window,
19                       "key-press-event",
20                        G_CALLBACK (on_key_press_event),
21                        search_bar);
22   }
```

Parameters

bar	a GtkSearchBar
event	a GdkEvent containing key press events

Returns

GDK_EVENT_STOP if the key press event resulted in text being entered in the search entry (and revealing the search bar if necessary), GDK_EVENT_PROPAGATE otherwise.

Since: 3.10

Types and Values

struct GtkSearchBar

```
struct GtkSearchBar;
```

---

struct GtkSearchBarClass

```
struct GtkSearchBarClass {
  GtkBinClass parent_class;
};
```

Members

Property Details

The `"search-mode-enabled"` property

`"search-mode-enabled"` `gboolean`

Whether the search mode is on and the search bar shown.

Flags: Read / Write

Default value: FALSE

The `"show-close-button"` property

`"show-close-button"` `gboolean`

Whether to show the close button in the toolbar.

Flags: Read / Write / Construct

Default value: FALSE

GtkEditable

GtkEditable — Interface for text-editing widgets

Functions

void	gtk_editable_select_region ()
gboolean	gtk_editable_get_selection_bounds ()
void	gtk_editable_insert_text ()
void	gtk_editable_delete_text ()
gchar *	gtk_editable_get_chars ()
void	gtk_editable_cut_clipboard ()
void	gtk_editable_copy_clipboard ()
void	gtk_editable_paste_clipboard ()
void	gtk_editable_delete_selection ()
void	gtk_editable_set_position ()
gint	gtk_editable_get_position ()
void	gtk_editable_set_editable ()
gboolean	gtk_editable_get_editable ()

Signals

void	changed	Run Last
void	delete-text	Run Last

void	insert-text	Run Last

Types and Values

GtkEditable

Object Hierarchy

```
GInterface
└── GtkEditable
```

Known Implementations

GtkEditable is implemented by GtkEntry, GtkSearchEntry and GtkSpinButton.

Includes

```
#include <gtk/gtk.h>
```

Description

The GtkEditable interface is an interface which should be implemented by text editing widgets, such as GtkEntry and GtkSpinButton. It contains functions for generically manipulating an editable widget, a large number of action signals used for key bindings, and several signals that an application can connect to to modify the behavior of a widget.

As an example of the latter usage, by connecting the following handler to "insert-text", an application can convert all entry into a widget into uppercase.

Forcing entry to uppercase.

```
1   #include <ctype.h>;
2
3   void
4   insert_text_handler (GtkEditable *editable,
5                        const gchar *text,
6                        gint         length,
7                        gint        *position,
8                        gpointer     data)
9   {
10    gchar *result = g_utf8_strup (text, length);
11
12    g_signal_handlers_block_by_func (editable,
13                                     (gpointer) insert_text_handler, data);
14    gtk_editable_insert_text (editable, result, length, position);
15    g_signal_handlers_unblock_by_func (editable,
16                                       (gpointer) insert_text_handler, data);
17
18    g_signal_stop_emission_by_name (editable, "insert_text");
19
20    g_free (result);
21  }
```

Functions

gtk_editable_select_region ()

```
void
gtk_editable_select_region (GtkEditable *editable,
                            gint start_pos,
                            gint end_pos);
```

Selects a region of text. The characters that are selected are those characters at positions from `start_pos` up to, but not including `end_pos` . If `end_pos` is negative, then the characters selected are those characters from `start_pos` to the end of the text.

Note that positions are specified in characters, not bytes.

[virtual set_selection_bounds]

Parameters

editable	a GtkEditable
start_pos	start of region
end_pos	end of region

---

gtk_editable_get_selection_bounds ()

```
gboolean
gtk_editable_get_selection_bounds (GtkEditable *editable,
                                   gint *start_pos,
                                   gint *end_pos);
```

Retrieves the selection bound of the editable. start_pos will be filled with the start of the selection and `end_pos` with end. If no text was selected both will be identical and `FALSE` will be returned.

Note that positions are specified in characters, not bytes.

Parameters

editable	a GtkEditable	
start_pos	location to store the starting position, or `NULL`.	[out][allow-none]
end_pos	location to store the end position, or `NULL`.	[out][allow-none]

Returns

`TRUE` if an area is selected, `FALSE` otherwise

---

gtk_editable_insert_text ()

```
void
gtk_editable_insert_text (GtkEditable *editable,
                          const gchar *new_text,
                          gint new_text_length,
                          gint *position);
```

Inserts `new_text_length` bytes of `new_text` into the contents of the widget, at position `position` .

Note that the position is in characters, not in bytes. The function updates `position` to point after the newly inserted text.

[virtual do_insert_text]

Parameters

editable	a GtkEditable	
new_text	the text to append	
new_text_length	the length of the text in bytes, or -1	
position	location of the position text will be inserted at.	[inout]

---

gtk_editable_delete_text ()

```
void
gtk_editable_delete_text (GtkEditable *editable,
                          gint start_pos,
                          gint end_pos);
```

Deletes a sequence of characters. The characters that are deleted are those characters at positions from `start_pos` up to, but not including `end_pos` . If `end_pos` is negative, then the characters deleted are those from `start_pos` to the end of the text.

Note that the positions are specified in characters, not bytes.

[virtual do_delete_text]

Parameters

editable	a GtkEditable
start_pos	start position
end_pos	end position

---

gtk_editable_get_chars ()

```
gchar *
gtk_editable_get_chars (GtkEditable *editable,
                        gint start_pos,
                        gint end_pos);
```

Retrieves a sequence of characters. The characters that are retrieved are those characters at positions from `start_pos` up to, but not including `end_pos` . If `end_pos` is negative, then the characters retrieved are those

characters from `start_pos` to the end of the text.

Note that positions are specified in characters, not bytes.

Parameters

editable	a GtkEditable
start_pos	start of text
end_pos	end of text

Returns

a pointer to the contents of the widget as a string. This string is allocated by the GtkEditable implementation and should be freed by the caller.

---

gtk_editable_cut_clipboard ()

```
void
gtk_editable_cut_clipboard (GtkEditable *editable);
```

Removes the contents of the currently selected content in the editable and puts it on the clipboard.

Parameters

editable	a GtkEditable

---

gtk_editable_copy_clipboard ()

```
void
gtk_editable_copy_clipboard (GtkEditable *editable);
```

Copies the contents of the currently selected content in the editable and puts it on the clipboard.

Parameters

editable	a GtkEditable

---

gtk_editable_paste_clipboard ()

```
void
gtk_editable_paste_clipboard (GtkEditable *editable);
```

Pastes the content of the clipboard to the current position of the cursor in the editable.

Parameters

editable	a GtkEditable

---

gtk_editable_delete_selection ()

```
void
gtk_editable_delete_selection (GtkEditable *editable);
```

Deletes the currently selected text of the editable. This call doesn’t do anything if there is no selected text.

Parameters

editable	a GtkEditable

---

gtk_editable_set_position ()

```
void
gtk_editable_set_position (GtkEditable *editable,
                           gint position);
```

Sets the cursor position in the editable to the given value.

The cursor is displayed before the character with the given (base 0) index in the contents of the editable. The value must be less than or equal to the number of characters in the editable. A value of -1 indicates that the position should be set after the last character of the editable. Note that `position` is in characters, not in bytes.

Parameters

editable	a GtkEditable
position	the position of the cursor

---

gtk_editable_get_position ()

```
gint
gtk_editable_get_position (GtkEditable *editable);
```

Retrieves the current position of the cursor relative to the start of the content of the editable.

Note that this position is in characters, not in bytes.

Parameters

editable	a GtkEditable

Returns

the cursor position

gtk_editable_set_editable ()

```
void
gtk_editable_set_editable (GtkEditable *editable,
                           gboolean is_editable);
```

Determines if the user can edit the text in the editable widget or not.

Parameters

editable	a GtkEditable
is_editable	TRUE if the user is allowed to edit the text in the widget

gtk_editable_get_editable ()

```
gboolean
gtk_editable_get_editable (GtkEditable *editable);
```

Retrieves whether editable is editable. See gtk_editable_set_editable().

Parameters

editable	a GtkEditable

Returns

TRUE if editable is editable.

Types and Values

GtkEditable

```
typedef struct _GtkEditable GtkEditable;
```

Signal Details

The "changed" signal

```
void
user_function (GtkEditable *editable,
               gpointer     user_data)
```

The ::changed signal is emitted at the end of a single user-visible operation on the contents of the GtkEditable.

E.g., a paste operation that replaces the contents of the selection will cause only one signal emission (even though it is implemented by first deleting the selection, then inserting the new content, and may cause multiple ::notify::text signals to be emitted).

Parameters

editable	the object which received the signal
user_data	user data set when the signal handler was connected.

Flags: Run Last

The `"delete-text"` signal

```
void
user_function (GtkEditable *editable,
               gint         start_pos,
               gint         end_pos,
               gpointer     user_data)
```

This signal is emitted when text is deleted from the widget by the user. The default handler for this signal will normally be responsible for deleting the text, so by connecting to this signal and then stopping the signal with `g_signal_stop_emission()`, it is possible to modify the range of deleted text, or prevent it from being deleted entirely. The `start_pos` and `end_pos` parameters are interpreted as for `gtk_editable_delete_text()`.

Parameters

editable	the object which received the signal
start_pos	the starting position
end_pos	the end position
user_data	user data set when the signal handler was connected.

Flags: Run Last

The `"insert-text"` signal

```
void
user_function (GtkEditable *editable,
               gchar       *new_text,
               gint         new_text_length,
               gpointer     position,
               gpointer     user_data)
```

This signal is emitted when text is inserted into the widget by the user. The default handler for this signal will normally be responsible for inserting the text, so by connecting to this signal and then stopping the signal with `g_signal_stop_emission()`, it is possible to modify the inserted text, or prevent it from being inserted entirely.

Parameters

editable	the object which received the signal	
new_text	the new text to insert	
new_text_length	the length of the new text, in bytes, or -1 if new_text is nul-terminated	
position	the position, in characters, at which to insert the new text. this is an in-out parameter. After the signal emission is finished, it should point after the newly inserted text.	[inout][type int]
user_data	user data set when the signal handler was connected.	

Flags: Run Last

Multiline Text Editor

Text Widget Overview — Overview of GtkTextBuffer, GtkTextView, and friends

GtkTextIter — Text buffer iterator

GtkTextMark — A position in the buffer preserved across buffer modifications

GtkTextBuffer — Stores attributed text for display in a GtkTextView

GtkTextTag — A tag that can be applied to text in a GtkTextBuffer

GtkTextTagTable — Collection of tags that can be used together

GtkTextView — Widget that displays a GtkTextBuffer

Text Widget Overview

Text Widget Overview — Overview of GtkTextBuffer, GtkTextView, and friends

Conceptual Overview

GTK+ has an extremely powerful framework for multiline text editing. The primary objects involved in the process are GtkTextBuffer, which represents the text being edited, and GtkTextView, a widget which can display a GtkTextBuffer. Each buffer can be displayed by any number of views.

One of the important things to remember about text in GTK+ is that it's in the UTF-8 encoding. This means that one character can be encoded as multiple bytes. Character counts are usually referred to as *offsets*, while byte counts are called *indexes*. If you confuse these two, things will work fine with ASCII, but as soon as your buffer contains multibyte characters, bad things will happen.

Text in a buffer can be marked with *tags*. A tag is an attribute that can be applied to some range of text. For example, a tag might be called "bold" and make the text inside the tag bold. However, the tag concept is more general than that; tags don't have to affect appearance. They can instead affect the behavior of mouse and key presses, "lock" a range of text so the user can't edit it, or countless other things. A tag is represented by a GtkTextTag object. One GtkTextTag can be applied to any number of text ranges in any number of buffers.

Each tag is stored in a GtkTextTagTable. A tag table defines a set of tags that can be used together. Each buffer has one tag table associated with it; only tags from that tag table can be used with the buffer. A single tag table can be shared between multiple buffers, however.

Tags can have names, which is convenient sometimes (for example, you can name your tag that makes things bold "bold"), but they can also be anonymous (which is convenient if you're creating tags on-the-fly).

Most text manipulation is accomplished with *iterators*, represented by a GtkTextIter. An iterator represents a position between two characters in the text buffer. GtkTextIter is a struct designed to be allocated on the stack; it's guaranteed to be copiable by value and never contain any heap-allocated data. Iterators are not valid indefinitely; whenever the buffer is modified in a way that affects the number of characters in the buffer, all outstanding iterators become invalid. (Note that deleting 5 characters and then reinserting 5 still invalidates iterators, though you end up with the same number of characters you pass through a state with a different number).

Because of this, iterators can't be used to preserve positions across buffer modifications. To preserve a position, the GtkTextMark object is ideal. You can think of a mark as an invisible cursor or insertion point; it floats in the buffer, saving a position. If the text surrounding the mark is deleted, the mark remains in the position the text once occupied; if text is inserted at the mark, the mark ends up either to the left or to the right of the new text, depending on its *gravity*. The standard text cursor in left-to-right languages is a mark with right gravity, because it stays to the right of inserted text.

Like tags, marks can be either named or anonymous. There are two marks built-in to GtkTextBuffer; these are named `"insert"` and `"selection_bound"` and refer to the insertion point and the boundary of the selection which is not the insertion point, respectively. If no text is selected, these two marks will be in the same position. You can manipulate what is selected and where the cursor appears by moving these marks around. [2]

Text buffers always contain at least one line, but may be empty (that is, buffers can contain zero characters). The last line in the text buffer never ends in a line separator (such as newline); the other lines in the buffer always end in a line separator. Line separators count as characters when computing character counts and character offsets. Note that some Unicode line separators are represented with multiple bytes in UTF-8, and the two-character sequence "\r\n" is also considered a line separator.

Simple Example

The simplest usage of GtkTextView might look like this:

```
1    GtkWidget *view;
2    GtkTextBuffer *buffer;
3
4    view = gtk_text_view_new ();
5
6    buffer = gtk_text_view_get_buffer (GTK_TEXT_VIEW (view));
7
8    gtk_text_buffer_set_text (buffer, "Hello, this is some text", -1);
9
10   /* Now you might put the view in a container and display it on the
11    * screen; when the user edits the text, signals on the buffer
12    * will be emitted, such as "changed", "insert_text", and so on.
13    */
```

In many cases it's also convenient to first create the buffer with `gtk_text_buffer_new()`, then create a widget for that buffer with `gtk_text_view_new_with_buffer()`. Or you can change the buffer the widget displays after the widget is created with `gtk_text_view_set_buffer()`.

Example of Changing Text Attributes

The way to affect text attributes in GtkTextView is to apply tags that change the attributes for a region of text. For text features that come from the theme — such as font and foreground color — use CSS to override their default values.

```
1    GtkWidget *view;
2    GtkTextBuffer *buffer;
3    GtkTextIter start, end;
4    PangoFontDescription *font_desc;
5    GdkRGBA rgba;
6    GtkTextTag *tag;
7    GtkCssProvider *provider;
8    GtkStyleContext *context;
9
10   view = gtk_text_view_new ();
11
12   buffer = gtk_text_view_get_buffer (GTK_TEXT_VIEW (view));
13
14   gtk_text_buffer_set_text (buffer, "Hello, this is some text", -1);
15
16   /* Change default font and color throughout the widget */
17   provider = gtk_css_provider_new ();
18   gtk_css_provider_load_from_data (provider,
19                                    "textview {"
20                                    " font: 15 serif;"
21                                    "  color: green;"
22                                    "}",
23                                    -1,
24                                    NULL);
25   context = gtk_widget_get_style_context (view);
26   gtk_style_context_add_provider (context,
27                                   GTK_STYLE_PROVIDER (provider),
28                                   GTK_STYLE_PROVIDER_PRIORITY_APPLICATION);
29
30   /* Change left margin throughout the widget */
31   gtk_text_view_set_left_margin (GTK_TEXT_VIEW (view), 30);
32
33   /* Use a tag to change the color for just one part of the widget */
34   tag = gtk_text_buffer_create_tag (buffer, "blue_foreground",
35                                     "foreground", "blue", NULL);
36   gtk_text_buffer_get_iter_at_offset (buffer, &start, 7);
37   gtk_text_buffer_get_iter_at_offset (buffer, &end, 12);
38   gtk_text_buffer_apply_tag (buffer, tag, &start, &end);
```

The gtk-demo application that comes with GTK+ contains more example code for GtkTextView.

---

[2] If you want to place the cursor in response to a user action, be sure to use `gtk_text_buffer_place_cursor()`, which moves both at once without causing a temporary selection (moving one then the other temporarily selects the range in between the old and new positions).

GtkTextIter

GtkTextIter — Text buffer iterator

Functions

GtkTextBuffer *	gtk_text_iter_get_buffer ()
GtkTextIter *	gtk_text_iter_copy ()
void	gtk_text_iter_assign ()
void	gtk_text_iter_free ()
gint	gtk_text_iter_get_offset ()
gint	gtk_text_iter_get_line ()
gint	gtk_text_iter_get_line_offset ()
gint	gtk_text_iter_get_line_index ()
gint	gtk_text_iter_get_visible_line_index ()
gint	gtk_text_iter_get_visible_line_offset ()
gunichar	gtk_text_iter_get_char ()
gchar *	gtk_text_iter_get_slice ()
gchar *	gtk_text_iter_get_text ()
gchar *	gtk_text_iter_get_visible_slice ()
gchar *	gtk_text_iter_get_visible_text ()
GdkPixbuf *	gtk_text_iter_get_pixbuf ()
GSList *	gtk_text_iter_get_marks ()
GSList *	gtk_text_iter_get_toggled_tags ()
GtkTextChildAnchor *	gtk_text_iter_get_child_anchor ()
gboolean	gtk_text_iter_starts_tag ()
gboolean	gtk_text_iter_begins_tag ()
gboolean	gtk_text_iter_ends_tag ()
gboolean	gtk_text_iter_toggles_tag ()
gboolean	gtk_text_iter_has_tag ()
GSList *	gtk_text_iter_get_tags ()
gboolean	gtk_text_iter_editable ()
gboolean	gtk_text_iter_can_insert ()
gboolean	gtk_text_iter_starts_word ()
gboolean	gtk_text_iter_ends_word ()
gboolean	gtk_text_iter_inside_word ()
gboolean	gtk_text_iter_starts_line ()
gboolean	gtk_text_iter_ends_line ()
gboolean	gtk_text_iter_starts_sentence ()
gboolean	gtk_text_iter_ends_sentence ()
gboolean	gtk_text_iter_inside_sentence ()
gboolean	gtk_text_iter_is_cursor_position ()
gint	gtk_text_iter_get_chars_in_line ()
gint	gtk_text_iter_get_bytes_in_line ()
gboolean	gtk_text_iter_get_attributes ()
PangoLanguage *	gtk_text_iter_get_language ()
gboolean	gtk_text_iter_is_end ()
gboolean	gtk_text_iter_is_start ()
gboolean	gtk_text_iter_forward_char ()
gboolean	gtk_text_iter_backward_char ()
gboolean	gtk_text_iter_forward_chars ()

gboolean	gtk_text_iter_backward_chars ()
gboolean	gtk_text_iter_forward_line ()
gboolean	gtk_text_iter_backward_line ()
gboolean	gtk_text_iter_forward_lines ()
gboolean	gtk_text_iter_backward_lines ()
gboolean	gtk_text_iter_forward_word_ends ()
gboolean	gtk_text_iter_backward_word_starts ()
gboolean	gtk_text_iter_forward_word_end ()
gboolean	gtk_text_iter_backward_word_start ()
gboolean	gtk_text_iter_forward_cursor_position ()
gboolean	gtk_text_iter_backward_cursor_position ()
gboolean	gtk_text_iter_forward_cursor_positions ()
gboolean	gtk_text_iter_backward_cursor_positions ()
gboolean	gtk_text_iter_backward_sentence_start ()
gboolean	gtk_text_iter_backward_sentence_starts ()
gboolean	gtk_text_iter_forward_sentence_end ()
gboolean	gtk_text_iter_forward_sentence_ends ()
gboolean	gtk_text_iter_forward_visible_word_ends ()
gboolean	gtk_text_iter_backward_visible_word_starts ()
gboolean	gtk_text_iter_forward_visible_word_end ()
gboolean	gtk_text_iter_backward_visible_word_start ()
gboolean	gtk_text_iter_forward_visible_cursor_position ()
gboolean	gtk_text_iter_backward_visible_cursor_position ()
gboolean	gtk_text_iter_forward_visible_cursor_positions ()
gboolean	gtk_text_iter_backward_visible_cursor_positions ()
gboolean	gtk_text_iter_forward_visible_line ()
gboolean	gtk_text_iter_backward_visible_line ()
gboolean	gtk_text_iter_forward_visible_lines ()
gboolean	gtk_text_iter_backward_visible_lines ()
void	gtk_text_iter_set_offset ()
void	gtk_text_iter_set_line ()
void	gtk_text_iter_set_line_offset ()
void	gtk_text_iter_set_line_index ()
void	gtk_text_iter_set_visible_line_index ()
void	gtk_text_iter_set_visible_line_offset ()
void	gtk_text_iter_forward_to_end ()
gboolean	gtk_text_iter_forward_to_line_end ()
gboolean	gtk_text_iter_forward_to_tag_toggle ()
gboolean	gtk_text_iter_backward_to_tag_toggle ()
gboolean	(*GtkTextCharPredicate) ()
gboolean	gtk_text_iter_forward_find_char ()
gboolean	gtk_text_iter_backward_find_char ()
gboolean	gtk_text_iter_forward_search ()
gboolean	gtk_text_iter_backward_search ()
gboolean	gtk_text_iter_equal ()
gint	gtk_text_iter_compare ()
gboolean	gtk_text_iter_in_range ()
void	gtk_text_iter_order ()

Types and Values

	GtkTextIter
enum	GtkTextSearchFlags

Object Hierarchy

```
GBoxed
└── GtkTextIter
```

Includes

```
#include <gtk/gtk.h>
```

Description

You may wish to begin by reading the text widget conceptual overview which gives an overview of all the objects and data types related to the text widget and how they work together.

Functions

gtk_text_iter_get_buffer ()

```
GtkTextBuffer *
gtk_text_iter_get_buffer (const GtkTextIter *iter);
```

Returns the GtkTextBuffer this iterator is associated with.

Parameters

iter	an iterator

Returns

the buffer.

[transfer none]

---

gtk_text_iter_copy ()

```
GtkTextIter *
gtk_text_iter_copy (const GtkTextIter *iter);
```

Creates a dynamically-allocated copy of an iterator. This function is not useful in applications, because iterators can be copied with a simple assignment (`GtkTextIter i = j;`). The function is used by language bindings.

Parameters

iter	an iterator

Returns

a copy of the `iter` , free with `gtk_text_iter_free()`

---

gtk_text_iter_assign ()

```
void
gtk_text_iter_assign (GtkTextIter *iter,
                      const GtkTextIter *other);
```

Assigns the value of `other` to `iter` . This function is not useful in applications, because iterators can be assigned with `GtkTextIter i = j;`. The function is used by language bindings.

Parameters

iter	a GtkTextIter
other	another GtkTextIter

Since: 3.2

---

gtk_text_iter_free ()

```
void
gtk_text_iter_free (GtkTextIter *iter);
```

Free an iterator allocated on the heap. This function is intended for use in language bindings, and is not especially useful for applications, because iterators can simply be allocated on the stack.

Parameters

iter	a dynamically-allocated iterator

---

gtk_text_iter_get_offset ()

```
gint
gtk_text_iter_get_offset (const GtkTextIter *iter);
```

Returns the character offset of an iterator. Each character in a GtkTextBuffer has an offset, starting with 0 for the first character in the buffer. Use `gtk_text_buffer_get_iter_at_offset()` to convert an offset back into an iterator.

Parameters

iter	an iterator

Returns

a character offset

---

gtk_text_iter_get_line ()

```
gint
gtk_text_iter_get_line (const GtkTextIter *iter);
```

Returns the line number containing the iterator. Lines in a GtkTextBuffer are numbered beginning with 0 for the first line in the buffer.

Parameters

iter	an iterator

Returns

a line number

---

gtk_text_iter_get_line_offset ()

```
gint
gtk_text_iter_get_line_offset (const GtkTextIter *iter);
```

Returns the character offset of the iterator, counting from the start of a newline-terminated line. The first character on the line has offset 0.

Parameters

iter	an iterator

Returns

offset from start of line

---

gtk_text_iter_get_line_index ()

```
gint
gtk_text_iter_get_line_index (const GtkTextIter *iter);
```

Returns the byte index of the iterator, counting from the start of a newline-terminated line. Remember that GtkTextBuffer encodes text in UTF-8, and that characters can require a variable number of bytes to represent.

Parameters

iter	an iterator

Returns

distance from start of line, in bytes

---

gtk_text_iter_get_visible_line_index ()

```
gint
gtk_text_iter_get_visible_line_index (const GtkTextIter *iter);
```

Returns the number of bytes from the start of the line to the given `iter` , not counting bytes that are invisible due to tags with the “invisible” flag toggled on.

Parameters

iter	a GtkTextIter

Returns

byte index of `iter` with respect to the start of the line

---

gtk_text_iter_get_visible_line_offset ()

```
gint
gtk_text_iter_get_visible_line_offset (const GtkTextIter *iter);
```

Returns the offset in characters from the start of the line to the given `iter` , not counting characters that are invisible due to tags with the “invisible” flag toggled on.

Parameters

iter	a GtkTextIter

Returns

offset in visible characters from the start of the line

---

gtk_text_iter_get_char ()

```
gunichar
gtk_text_iter_get_char (const GtkTextIter *iter);
```

The Unicode character at this iterator is returned. (Equivalent to operator* on a C++ iterator.) If the element at this iterator is a non-character element, such as an image embedded in the buffer, the Unicode “unknown”

character 0xFFFC is returned. If invoked on the end iterator, zero is returned; zero is not a valid Unicode character. So you can write a loop which ends when `gtk_text_iter_get_char()` returns 0.

Parameters

iter	an iterator

Returns

a Unicode character, or 0 if `iter` is not dereferenceable

---

gtk_text_iter_get_slice ()

```
gchar *
gtk_text_iter_get_slice (const GtkTextIter *start,
                         const GtkTextIter *end);
```

Returns the text in the given range. A “slice” is an array of characters encoded in UTF-8 format, including the Unicode “unknown” character 0xFFFC for iterable non-character elements in the buffer, such as images. Because images are encoded in the slice, byte and character offsets in the returned array will correspond to byte offsets in the text buffer. Note that 0xFFFC can occur in normal text as well, so it is not a reliable indicator that a pixbuf or widget is in the buffer.

Parameters

start	iterator at start of a range
end	iterator at end of a range

Returns

slice of text from the buffer.

[transfer full]

---

gtk_text_iter_get_text ()

```
gchar *
gtk_text_iter_get_text (const GtkTextIter *start,
                        const GtkTextIter *end);
```

Returns text in the given range. If the range contains non-text elements such as images, the character and byte offsets in the returned string will not correspond to character and byte offsets in the buffer. If you want offsets to correspond, see `gtk_text_iter_get_slice()`.

Parameters

start	iterator at start of a range
end	iterator at end of a range

Returns

array of characters from the buffer.

[transfer full]

---

gtk_text_iter_get_visible_slice ()

```
gchar *
gtk_text_iter_get_visible_slice (const GtkTextIter *start,
                                 const GtkTextIter *end);
```

Like `gtk_text_iter_get_slice()`, but invisible text is not included. Invisible text is usually invisible because a GtkTextTag with the “invisible” attribute turned on has been applied to it.

Parameters

start	iterator at start of range
end	iterator at end of range

Returns

slice of text from the buffer.

[transfer full]

---

gtk_text_iter_get_visible_text ()

```
gchar *
gtk_text_iter_get_visible_text (const GtkTextIter *start,
                                const GtkTextIter *end);
```

Like `gtk_text_iter_get_text()`, but invisible text is not included. Invisible text is usually invisible because a GtkTextTag with the “invisible” attribute turned on has been applied to it.

Parameters

start	iterator at start of range
end	iterator at end of range

Returns

string containing visible text in the range.

[transfer full]

---

gtk_text_iter_get_pixbuf ()

```
GdkPixbuf *
gtk_text_iter_get_pixbuf (const GtkTextIter *iter);
```

If the element at `iter` is a pixbuf, the pixbuf is returned (with no new reference count added). Otherwise, `NULL` is returned.

Parameters

iter	an iterator

Returns

the pixbuf at `iter` .

[transfer none]

---

gtk_text_iter_get_marks ()

```
GSList *
gtk_text_iter_get_marks (const GtkTextIter *iter);
```

Returns a list of all GtkTextMark at this location. Because marks are not iterable (they don’t take up any "space" in the buffer, they are just marks in between iterable locations), multiple marks can exist in the same place. The returned list is not in any meaningful order.

Parameters

iter	an iterator

Returns

list of GtkTextMark.

[element-type GtkTextMark][transfer container]

---

gtk_text_iter_get_toggled_tags ()

```
GSList *
gtk_text_iter_get_toggled_tags (const GtkTextIter *iter,
                                gboolean toggled_on);
```

Returns a list of GtkTextTag that are toggled on or off at this point. (If `toggled_on` is `TRUE`, the list contains tags that are toggled on.) If a tag is toggled on at `iter` , then some non-empty range of characters following `iter` has that tag applied to it. If a tag is toggled off, then some non-empty range following `iter` does not have the tag applied to it.

Parameters

iter	an iterator
toggled_on	`TRUE` to get toggled-on tags

Returns

tags toggled at this point.

[element-type GtkTextTag][transfer container]

---

gtk_text_iter_get_child_anchor ()

```
GtkTextChildAnchor *
gtk_text_iter_get_child_anchor (const GtkTextIter *iter);
```

If the location at `iter` contains a child anchor, the anchor is returned (with no new reference count added). Otherwise, `NULL` is returned.

Parameters

iter	an iterator

Returns

the anchor at `iter` .

[transfer none]

---

gtk_text_iter_starts_tag ()

```
gboolean
gtk_text_iter_starts_tag (const GtkTextIter *iter,
                          GtkTextTag *tag);
```

Returns `TRUE` if `tag` is toggled on at exactly this point. If `tag` is `NULL`, returns `TRUE` if any tag is toggled on at this point.

Note that if `gtk_text_iter_starts_tag()` returns `TRUE`, it means that `iter` is at the beginning of the tagged range, and that the character at `iter` is inside the tagged range. In other words, unlike `gtk_text_iter_ends_tag()`, if `gtk_text_iter_starts_tag()` returns `TRUE`, `gtk_text_iter_has_tag()` will also return `TRUE` for the same parameters.

Parameters

iter	an iterator	
tag	a GtkTextTag, or `NULL`.	[nullable]

Returns

whether `iter` is the start of a range tagged with `tag`

Since: 3.20

---

gtk_text_iter_begins_tag ()

```
gboolean
gtk_text_iter_begins_tag (const GtkTextIter *iter,
                          GtkTextTag *tag);
```

`gtk_text_iter_begins_tag` has been deprecated since version 3.20 and should not be used in newly-written code.

Use `gtk_text_iter_starts_tag()` instead.

Returns `TRUE` if `tag` is toggled on at exactly this point. If `tag` is `NULL`, returns `TRUE` if any tag is toggled on at this point.

Note that if `gtk_text_iter_begins_tag()` returns `TRUE`, it means that `iter` is at the beginning of the tagged range, and that the character at `iter` is inside the tagged range. In other words, unlike `gtk_text_iter_ends_tag()`, if `gtk_text_iter_begins_tag()` returns `TRUE`, `gtk_text_iter_has_tag()` will also return `TRUE` for the same parameters.

Parameters

iter	an iterator	
tag	a GtkTextTag, or `NULL`.	[nullable]

Returns

whether `iter` is the start of a range tagged with `tag`

---

gtk_text_iter_ends_tag ()

```
gboolean
gtk_text_iter_ends_tag (const GtkTextIter *iter,
                        GtkTextTag *tag);
```

Returns `TRUE` if `tag` is toggled off at exactly this point. If `tag` is `NULL`, returns `TRUE` if any tag is toggled off at this point.

Note that if `gtk_text_iter_ends_tag()` returns `TRUE`, it means that `iter` is at the end of the tagged range, but that the character at `iter` is outside the tagged range. In other words, unlike `gtk_text_iter_starts_tag()`, if `gtk_text_iter_ends_tag()` returns `TRUE`, `gtk_text_iter_has_tag()` will return `FALSE` for the same parameters.

Parameters

iter	an iterator

tag	a GtkTextTag, or NULL.	[allow-none]

Returns

whether iter is the end of a range tagged with tag

---

gtk_text_iter_toggles_tag ()

```
gboolean
gtk_text_iter_toggles_tag (const GtkTextIter *iter,
                           GtkTextTag *tag);
```

This is equivalent to (gtk_text_iter_starts_tag() || gtk_text_iter_ends_tag()), i.e. it tells you whether a range with tag applied to it begins or ends at iter .

Parameters

iter	an iterator	
tag	a GtkTextTag, or NULL.	[allow-none]

Returns

whether tag is toggled on or off at iter

---

gtk_text_iter_has_tag ()

```
gboolean
gtk_text_iter_has_tag (const GtkTextIter *iter,
                       GtkTextTag *tag);
```

Returns TRUE if iter points to a character that is part of a range tagged with tag . See also gtk_text_iter_starts_tag() and gtk_text_iter_ends_tag().

Parameters

iter	an iterator
tag	a GtkTextTag

Returns

whether iter is tagged with tag

---

gtk_text_iter_get_tags ()

```
GSList *
gtk_text_iter_get_tags (const GtkTextIter *iter);
```

Returns a list of tags that apply to `iter` , in ascending order of priority (highest-priority tags are last). The GtkTextTag in the list don’t have a reference added, but you have to free the list itself.

Parameters

iter	a GtkTextIter

Returns

list of GtkTextTag.

[element-type GtkTextTag][transfer container]

---

gtk_text_iter_editable ()

```
gboolean
gtk_text_iter_editable (const GtkTextIter *iter,
                        gboolean default_setting);
```

Returns whether the character at `iter` is within an editable region of text. Non-editable text is “locked” and can’t be changed by the user via GtkTextView. This function is simply a convenience wrapper around `gtk_text_iter_get_attributes()`. If no tags applied to this text affect editability, `default_setting` will be returned.

You don’t want to use this function to decide whether text can be inserted at `iter` , because for insertion you don’t want to know whether the char at `iter` is inside an editable range, you want to know whether a new character inserted at `iter` would be inside an editable range. Use `gtk_text_iter_can_insert()` to handle this case.

Parameters

iter	an iterator
default_setting	`TRUE` if text is editable by default

Returns

whether `iter` is inside an editable range

---

gtk_text_iter_can_insert ()

```
gboolean
gtk_text_iter_can_insert (const GtkTextIter *iter,
                          gboolean default_editability);
```

Considering the default editability of the buffer, and tags that affect editability, determines whether text inserted at `iter` would be editable. If text inserted at `iter` would be editable then the user should be allowed to insert text at `iter` . `gtk_text_buffer_insert_interactive()` uses this function to decide whether insertions are allowed at a given position.

Parameters

iter	an iterator
default_editability	`TRUE` if text is editable by default

Returns

whether text inserted at `iter` would be editable

---

gtk_text_iter_starts_word ()

```
gboolean
gtk_text_iter_starts_word (const GtkTextIter *iter);
```

Determines whether `iter` begins a natural-language word. Word breaks are determined by Pango and should be correct for nearly any language (if not, the correct fix would be to the Pango word break algorithms).

Parameters

iter	a GtkTextIter

Returns

`TRUE` if `iter` is at the start of a word

---

gtk_text_iter_ends_word ()

```
gboolean
gtk_text_iter_ends_word (const GtkTextIter *iter);
```

Determines whether `iter` ends a natural-language word. Word breaks are determined by Pango and should be correct for nearly any language (if not, the correct fix would be to the Pango word break algorithms).

Parameters

iter	a GtkTextIter

Returns

`TRUE` if `iter` is at the end of a word

---

gtk_text_iter_inside_word ()

```
gboolean
```

```
gtk_text_iter_inside_word (const GtkTextIter *iter);
```
Determines whether the character pointed by `iter` is part of a natural-language word (as opposed to say inside some whitespace). Word breaks are determined by Pango and should be correct for nearly any language (if not, the correct fix would be to the Pango word break algorithms).

Note that if `gtk_text_iter_starts_word()` returns `TRUE`, then this function returns `TRUE` too, since `iter` points to the first character of the word.

Parameters

iter	a GtkTextIter

Returns

`TRUE` if `iter` is inside a word

---

gtk_text_iter_starts_line ()

```
gboolean
gtk_text_iter_starts_line (const GtkTextIter *iter);
```
Returns `TRUE` if `iter` begins a paragraph, i.e. if `gtk_text_iter_get_line_offset()` would return 0. However this function is potentially more efficient than `gtk_text_iter_get_line_offset()` because it doesn't have to compute the offset, it just has to see whether it's 0.

Parameters

iter	an iterator

Returns

whether `iter` begins a line

---

gtk_text_iter_ends_line ()

```
gboolean
gtk_text_iter_ends_line (const GtkTextIter *iter);
```
Returns `TRUE` if `iter` points to the start of the paragraph delimiter characters for a line (delimiters will be either a newline, a carriage return, a carriage return followed by a newline, or a Unicode paragraph separator character). Note that an iterator pointing to the \n of a \r\n pair will not be counted as the end of a line, the line ends before the \r. The end iterator is considered to be at the end of a line, even though there are no paragraph delimiter chars there.

Parameters

iter	an iterator

Returns

whether `iter` is at the end of a line

---

gtk_text_iter_starts_sentence ()

```
gboolean
gtk_text_iter_starts_sentence (const GtkTextIter *iter);
```

Determines whether `iter` begins a sentence. Sentence boundaries are determined by Pango and should be correct for nearly any language (if not, the correct fix would be to the Pango text boundary algorithms).

Parameters

iter	a GtkTextIter

Returns

`TRUE` if `iter` is at the start of a sentence.

---

gtk_text_iter_ends_sentence ()

```
gboolean
gtk_text_iter_ends_sentence (const GtkTextIter *iter);
```

Determines whether `iter` ends a sentence. Sentence boundaries are determined by Pango and should be correct for nearly any language (if not, the correct fix would be to the Pango text boundary algorithms).

Parameters

iter	a GtkTextIter

Returns

`TRUE` if `iter` is at the end of a sentence.

---

gtk_text_iter_inside_sentence ()

```
gboolean
gtk_text_iter_inside_sentence (const GtkTextIter *iter);
```

Determines whether `iter` is inside a sentence (as opposed to in between two sentences, e.g. after a period and before the first letter of the next sentence). Sentence boundaries are determined by Pango and should be correct for nearly any language (if not, the correct fix would be to the Pango text boundary algorithms).

Parameters

iter	a GtkTextIter

Returns

`TRUE` if `iter` is inside a sentence.

---

gtk_text_iter_is_cursor_position ()

```
gboolean
gtk_text_iter_is_cursor_position (const GtkTextIter *iter);
```

See `gtk_text_iter_forward_cursor_position()` or PangoLogAttr or `pango_break()` for details on what a cursor position is.

Parameters

iter	a GtkTextIter

Returns

`TRUE` if the cursor can be placed at `iter`

---

gtk_text_iter_get_chars_in_line ()

```
gint
gtk_text_iter_get_chars_in_line (const GtkTextIter *iter);
```

Returns the number of characters in the line containing `iter` , including the paragraph delimiters.

Parameters

iter	an iterator

Returns

number of characters in the line

---

gtk_text_iter_get_bytes_in_line ()

```
gint
gtk_text_iter_get_bytes_in_line (const GtkTextIter *iter);
```

Returns the number of bytes in the line containing `iter` , including the paragraph delimiters.

Parameters

iter	an iterator

Returns

number of bytes in the line

---

gtk_text_iter_get_attributes ()

```
gboolean
gtk_text_iter_get_attributes (const GtkTextIter *iter,
                              GtkTextAttributes *values);
```

Computes the effect of any tags applied to this spot in the text. The `values` parameter should be initialized to the default settings you wish to use if no tags are in effect. You'd typically obtain the defaults from `gtk_text_view_get_default_attributes()`.

gtk_text_iter_get_attributes() will modify `values` , applying the effects of any tags present at `iter` . If any tags affected `values` , the function returns `TRUE`.

Parameters

iter	an iterator	
values	a GtkTextAttributes to be filled in.	[out]

Returns

`TRUE` if `values` was modified

---

gtk_text_iter_get_language ()

```
PangoLanguage *
gtk_text_iter_get_language (const GtkTextIter *iter);
```

A convenience wrapper around `gtk_text_iter_get_attributes()`, which returns the language in effect at `iter` . If no tags affecting language apply to `iter` , the return value is identical to that of `gtk_get_default_language()`.

Parameters

iter	an iterator

Returns

language in effect at `iter` .

[transfer full]

---

gtk_text_iter_is_end ()

```
gboolean
gtk_text_iter_is_end (const GtkTextIter *iter);
```

Returns `TRUE` if `iter` is the end iterator, i.e. one past the last dereferenceable iterator in the buffer. `gtk_text_iter_is_end()` is the most efficient way to check whether an iterator is the end iterator.

Parameters

iter	an iterator

Returns

whether `iter` is the end iterator

---

gtk_text_iter_is_start ()

```
gboolean
gtk_text_iter_is_start (const GtkTextIter *iter);
```

Returns `TRUE` if `iter` is the first iterator in the buffer, that is if `iter` has a character offset of 0.

Parameters

iter	an iterator

Returns

whether `iter` is the first in the buffer

---

gtk_text_iter_forward_char ()

```
gboolean
gtk_text_iter_forward_char (GtkTextIter *iter);
```

Moves `iter` forward by one character offset. Note that images embedded in the buffer occupy 1 character slot, so `gtk_text_iter_forward_char()` may actually move onto an image instead of a character, if you have images in your buffer. If `iter` is the end iterator or one character before it, `iter` will now point at the end iterator, and `gtk_text_iter_forward_char()` returns `FALSE` for convenience when writing loops.

Parameters

iter	an iterator

Returns

whether `iter` moved and is dereferenceable

---

gtk_text_iter_backward_char ()

```
gboolean
gtk_text_iter_backward_char (GtkTextIter *iter);
```

Moves backward by one character offset. Returns `TRUE` if movement was possible; if `iter` was the first in the buffer (character offset 0), `gtk_text_iter_backward_char()` returns `FALSE` for convenience when writing loops.

Parameters

iter	an iterator

Returns

whether movement was possible

---

gtk_text_iter_forward_chars ()

```
gboolean
gtk_text_iter_forward_chars (GtkTextIter *iter,
                             gint count);
```

Moves `count` characters if possible (if `count` would move past the start or end of the buffer, moves to the start or end of the buffer). The return value indicates whether the new position of `iter` is different from its original position, and dereferenceable (the last iterator in the buffer is not dereferenceable). If `count` is 0, the function does nothing and returns `FALSE`.

Parameters

iter	an iterator
count	number of characters to move, may be negative

Returns

whether `iter` moved and is dereferenceable

---

gtk_text_iter_backward_chars ()

```
gboolean
gtk_text_iter_backward_chars (GtkTextIter *iter,
                              gint count);
```

Moves `count` characters backward, if possible (if `count` would move past the start or end of the buffer, moves to the start or end of the buffer). The return value indicates whether the iterator moved onto a dereferenceable position; if the iterator didn't move, or moved onto the end iterator, then `FALSE` is returned. If `count` is 0, the function does nothing and returns `FALSE`.

Parameters

iter	an iterator
count	number of characters to move

Returns

whether `iter` moved and is dereferenceable

---

gtk_text_iter_forward_line ()

```
gboolean
gtk_text_iter_forward_line (GtkTextIter *iter);
```

Moves `iter` to the start of the next line. If the iter is already on the last line of the buffer, moves the iter to the end of the current line. If after the operation, the iter is at the end of the buffer and not dereferencable, returns `FALSE`. Otherwise, returns `TRUE`.

Parameters

iter	an iterator

Returns

whether `iter` can be dereferenced

---

gtk_text_iter_backward_line ()

```
gboolean
gtk_text_iter_backward_line (GtkTextIter *iter);
```

Moves `iter` to the start of the previous line. Returns `TRUE` if `iter` could be moved; i.e. if `iter` was at character offset 0, this function returns `FALSE`. Therefore if `iter` was already on line 0, but not at the start of the line, `iter` is snapped to the start of the line and the function returns `TRUE`. (Note that this implies that in a loop calling this function, the line number may not change on every iteration, if your first iteration is on line 0.)

Parameters

iter	an iterator

Returns

whether `iter` moved

---

gtk_text_iter_forward_lines ()

```
gboolean
gtk_text_iter_forward_lines (GtkTextIter *iter,
                             gint count);
```

Moves `count` lines forward, if possible (if `count` would move past the start or end of the buffer, moves to the start or end of the buffer). The return value indicates whether the iterator moved onto a dereferenceable position; if the iterator didn't move, or moved onto the end iterator, then `FALSE` is returned. If `count` is 0, the function does nothing and returns `FALSE`. If `count` is negative, moves backward by 0 - `count` lines.

Parameters

iter	a GtkTextIter
count	number of lines to move forward

Returns

whether `iter` moved and is dereferenceable

---

gtk_text_iter_backward_lines ()

```
gboolean
gtk_text_iter_backward_lines (GtkTextIter *iter,
                              gint count);
```

Moves `count` lines backward, if possible (if `count` would move past the start or end of the buffer, moves to the start or end of the buffer). The return value indicates whether the iterator moved onto a dereferenceable position; if the iterator didn't move, or moved onto the end iterator, then `FALSE` is returned. If `count` is 0, the function does nothing and returns `FALSE`. If `count` is negative, moves forward by 0 - `count` lines.

Parameters

iter	a GtkTextIter
count	number of lines to move backward

Returns

whether `iter` moved and is dereferenceable

---

gtk_text_iter_forward_word_ends ()

```
gboolean
```

```
gtk_text_iter_forward_word_ends (GtkTextIter *iter,
                                 gint count);
```

Calls `gtk_text_iter_forward_word_end()` up to `count` times.

Parameters

iter	a GtkTextIter
count	number of times to move

Returns

`TRUE` if `iter` moved and is not the end iterator

---

gtk_text_iter_backward_word_starts ()

```
gboolean
gtk_text_iter_backward_word_starts (GtkTextIter *iter,
                                    gint count);
```

Calls `gtk_text_iter_backward_word_start()` up to `count` times.

Parameters

iter	a GtkTextIter
count	number of times to move

Returns

`TRUE` if `iter` moved and is not the end iterator

---

gtk_text_iter_forward_word_end ()

```
gboolean
gtk_text_iter_forward_word_end (GtkTextIter *iter);
```

Moves forward to the next word end. (If `iter` is currently on a word end, moves forward to the next one after that.) Word breaks are determined by Pango and should be correct for nearly any language (if not, the correct fix would be to the Pango word break algorithms).

Parameters

iter	a GtkTextIter

Returns

`TRUE` if `iter` moved and is not the end iterator

---

gtk_text_iter_backward_word_start ()

```
gboolean
gtk_text_iter_backward_word_start (GtkTextIter *iter);
```

Moves backward to the previous word start. (If `iter` is currently on a word start, moves backward to the next one after that.) Word breaks are determined by Pango and should be correct for nearly any language (if not, the correct fix would be to the Pango word break algorithms).

Parameters

iter	a GtkTextIter

Returns

`TRUE` if `iter` moved and is not the end iterator

---

gtk_text_iter_forward_cursor_position ()

```
gboolean
gtk_text_iter_forward_cursor_position (GtkTextIter *iter);
```

Moves `iter` forward by a single cursor position. Cursor positions are (unsurprisingly) positions where the cursor can appear. Perhaps surprisingly, there may not be a cursor position between all characters. The most common example for European languages would be a carriage return/newline sequence. For some Unicode characters, the equivalent of say the letter “a” with an accent mark will be represented as two characters, first the letter then a "combining mark" that causes the accent to be rendered; so the cursor can’t go between those two characters. See also the PangoLogAttr and `pango_break()` function.

Parameters

iter	a GtkTextIter

Returns

`TRUE` if we moved and the new position is dereferenceable

---

gtk_text_iter_backward_cursor_position ()

```
gboolean
gtk_text_iter_backward_cursor_position
                                (GtkTextIter *iter);
```

Like `gtk_text_iter_forward_cursor_position()`, but moves backward.

Parameters

iter	a GtkTextIter

Returns

TRUE if we moved

---

gtk_text_iter_forward_cursor_positions ()

```
gboolean
gtk_text_iter_forward_cursor_positions
                                (GtkTextIter *iter,
                                 gint count);
```

Moves up to count cursor positions. See gtk_text_iter_forward_cursor_position() for details.

Parameters

iter	a GtkTextIter
count	number of positions to move

Returns

TRUE if we moved and the new position is dereferenceable

---

gtk_text_iter_backward_cursor_positions ()

```
gboolean
gtk_text_iter_backward_cursor_positions
                                (GtkTextIter *iter,
                                 gint count);
```

Moves up to count cursor positions. See gtk_text_iter_forward_cursor_position() for details.

Parameters

iter	a GtkTextIter
count	number of positions to move

Returns

TRUE if we moved and the new position is dereferenceable

---

gtk_text_iter_backward_sentence_start ()

```
gboolean
```

```
gtk_text_iter_backward_sentence_start (GtkTextIter *iter);
```
Moves backward to the previous sentence start; if `iter` is already at the start of a sentence, moves backward to the next one. Sentence boundaries are determined by Pango and should be correct for nearly any language (if not, the correct fix would be to the Pango text boundary algorithms).

Parameters

iter	a GtkTextIter

Returns

`TRUE` if `iter` moved and is not the end iterator

---

gtk_text_iter_backward_sentence_starts ()

```
gboolean
gtk_text_iter_backward_sentence_starts
                                (GtkTextIter *iter,
                                 gint count);
```
Calls `gtk_text_iter_backward_sentence_start()` up to `count` times, or until it returns `FALSE`. If `count` is negative, moves forward instead of backward.

Parameters

iter	a GtkTextIter
count	number of sentences to move

Returns

`TRUE` if `iter` moved and is not the end iterator

---

gtk_text_iter_forward_sentence_end ()

```
gboolean
gtk_text_iter_forward_sentence_end (GtkTextIter *iter);
```
Moves forward to the next sentence end. (If `iter` is at the end of a sentence, moves to the next end of sentence.) Sentence boundaries are determined by Pango and should be correct for nearly any language (if not, the correct fix would be to the Pango text boundary algorithms).

Parameters

iter	a GtkTextIter

Returns

TRUE if iter moved and is not the end iterator

---

gtk_text_iter_forward_sentence_ends ()

```
gboolean
gtk_text_iter_forward_sentence_ends (GtkTextIter *iter,
                                     gint count);
```

Calls gtk_text_iter_forward_sentence_end() count times (or until gtk_text_iter_forward_sentence_end() returns FALSE). If count is negative, moves backward instead of forward.

Parameters

iter	a GtkTextIter
count	number of sentences to move

Returns

TRUE if iter moved and is not the end iterator

---

gtk_text_iter_forward_visible_word_ends ()

```
gboolean
gtk_text_iter_forward_visible_word_ends
                                (GtkTextIter *iter,
                                 gint count);
```

Calls gtk_text_iter_forward_visible_word_end() up to count times.

Parameters

iter	a GtkTextIter
count	number of times to move

Returns

TRUE if iter moved and is not the end iterator

Since: 2.4

---

gtk_text_iter_backward_visible_word_starts ()

```
gboolean
gtk_text_iter_backward_visible_word_starts
                                (GtkTextIter *iter,
```

```
                                gint count);
```

Calls `gtk_text_iter_backward_visible_word_start()` up to `count` times.

Parameters

iter	a GtkTextIter
count	number of times to move

Returns

`TRUE` if `iter` moved and is not the end iterator

Since: 2.4

---

gtk_text_iter_forward_visible_word_end ()

```
gboolean
gtk_text_iter_forward_visible_word_end
                                (GtkTextIter *iter);
```

Moves forward to the next visible word end. (If `iter` is currently on a word end, moves forward to the next one after that.) Word breaks are determined by Pango and should be correct for nearly any language (if not, the correct fix would be to the Pango word break algorithms).

Parameters

iter	a GtkTextIter

Returns

`TRUE` if `iter` moved and is not the end iterator

Since: 2.4

---

gtk_text_iter_backward_visible_word_start ()

```
gboolean
gtk_text_iter_backward_visible_word_start
                                (GtkTextIter *iter);
```

Moves backward to the previous visible word start. (If `iter` is currently on a word start, moves backward to the next one after that.) Word breaks are determined by Pango and should be correct for nearly any language (if not, the correct fix would be to the Pango word break algorithms).

Parameters

iter	a GtkTextIter

Returns

`TRUE` if `iter` moved and is not the end iterator

Since: 2.4

---

gtk_text_iter_forward_visible_cursor_position ()

```
gboolean
gtk_text_iter_forward_visible_cursor_position
                                (GtkTextIter *iter);
```

Moves `iter` forward to the next visible cursor position. See `gtk_text_iter_forward_cursor_position()` for details.

Parameters

iter	a GtkTextIter

Returns

`TRUE` if we moved and the new position is dereferenceable

Since: 2.4

---

gtk_text_iter_backward_visible_cursor_position ()

```
gboolean
gtk_text_iter_backward_visible_cursor_position
                                (GtkTextIter *iter);
```

Moves `iter` forward to the previous visible cursor position. See `gtk_text_iter_backward_cursor_position()` for details.

Parameters

iter	a GtkTextIter

Returns

`TRUE` if we moved and the new position is dereferenceable

Since: 2.4

---

gtk_text_iter_forward_visible_cursor_positions ()

```
gboolean
gtk_text_iter_forward_visible_cursor_positions
```

```
                                (GtkTextIter *iter,
                                 gint count);
```

Moves up to `count` visible cursor positions. See `gtk_text_iter_forward_cursor_position()` for details.

Parameters

iter	a GtkTextIter
count	number of positions to move

Returns

`TRUE` if we moved and the new position is dereferenceable

Since: 2.4

---

gtk_text_iter_backward_visible_cursor_positions ()

```
gboolean
gtk_text_iter_backward_visible_cursor_positions
                                (GtkTextIter *iter,
                                 gint count);
```

Moves up to `count` visible cursor positions. See `gtk_text_iter_backward_cursor_position()` for details.

Parameters

iter	a GtkTextIter
count	number of positions to move

Returns

`TRUE` if we moved and the new position is dereferenceable

Since: 2.4

---

gtk_text_iter_forward_visible_line ()

```
gboolean
gtk_text_iter_forward_visible_line (GtkTextIter *iter);
```

Moves `iter` to the start of the next visible line. Returns `TRUE` if there was a next line to move to, and `FALSE` if `iter` was simply moved to the end of the buffer and is now not dereferenceable, or if `iter` was already at the end of the buffer.

Parameters

iter	an iterator

Returns

whether `iter` can be dereferenced

Since: 2.8

---

gtk_text_iter_backward_visible_line ()

```
gboolean
gtk_text_iter_backward_visible_line (GtkTextIter *iter);
```

Moves `iter` to the start of the previous visible line. Returns `TRUE` if `iter` could be moved; i.e. if `iter` was at character offset 0, this function returns `FALSE`. Therefore if `iter` was already on line 0, but not at the start of the line, `iter` is snapped to the start of the line and the function returns `TRUE`. (Note that this implies that in a loop calling this function, the line number may not change on every iteration, if your first iteration is on line 0.)

Parameters

iter	an iterator

Returns

whether `iter` moved

Since: 2.8

---

gtk_text_iter_forward_visible_lines ()

```
gboolean
gtk_text_iter_forward_visible_lines (GtkTextIter *iter,
                                     gint count);
```

Moves `count` visible lines forward, if possible (if `count` would move past the start or end of the buffer, moves to the start or end of the buffer). The return value indicates whether the iterator moved onto a dereferenceable position; if the iterator didn't move, or moved onto the end iterator, then `FALSE` is returned. If `count` is 0, the function does nothing and returns `FALSE`. If `count` is negative, moves backward by 0 - `count` lines.

Parameters

iter	a GtkTextIter
count	number of lines to move forward

Returns

whether `iter` moved and is dereferenceable

Since: 2.8

---

gtk_text_iter_backward_visible_lines ()

```
gboolean
gtk_text_iter_backward_visible_lines (GtkTextIter *iter,
                                      gint count);
```

Moves `count` visible lines backward, if possible (if `count` would move past the start or end of the buffer, moves to the start or end of the buffer). The return value indicates whether the iterator moved onto a dereferenceable position; if the iterator didn't move, or moved onto the end iterator, then `FALSE` is returned. If `count` is 0, the function does nothing and returns `FALSE`. If `count` is negative, moves forward by 0 - `count` lines.

Parameters

iter	a GtkTextIter
count	number of lines to move backward

Returns

whether `iter` moved and is dereferenceable

Since: 2.8

---

gtk_text_iter_set_offset ()

```
void
gtk_text_iter_set_offset (GtkTextIter *iter,
                          gint char_offset);
```

Sets `iter` to point to `char_offset` . `char_offset` counts from the start of the entire text buffer, starting with 0.

Parameters

iter	a GtkTextIter
char_offset	a character number

---

gtk_text_iter_set_line ()

```
void
gtk_text_iter_set_line (GtkTextIter *iter,
                        gint line_number);
```

Moves iterator `iter` to the start of the line `line_number` . If `line_number` is negative or larger than the number of lines in the buffer, moves `iter` to the start of the last line in the buffer.

Parameters

iter	a GtkTextIter
line_number	line number (counted from 0)

---

gtk_text_iter_set_line_offset ()

```
void
gtk_text_iter_set_line_offset (GtkTextIter *iter,
                               gint char_on_line);
```

Moves `iter` within a line, to a new character (not byte) offset. The given character offset must be less than or equal to the number of characters in the line; if equal, `iter` moves to the start of the next line. See `gtk_text_iter_set_line_index()` if you have a byte index rather than a character offset.

Parameters

iter	a GtkTextIter
char_on_line	a character offset relative to the start of `iter` 's current line

---

gtk_text_iter_set_line_index ()

```
void
gtk_text_iter_set_line_index (GtkTextIter *iter,
                              gint byte_on_line);
```

Same as `gtk_text_iter_set_line_offset()`, but works with a byte index. The given byte index must be at the start of a character, it can't be in the middle of a UTF-8 encoded character.

Parameters

iter	a GtkTextIter
byte_on_line	a byte index relative to the start of `iter` 's current line

---

gtk_text_iter_set_visible_line_index ()

```
void
gtk_text_iter_set_visible_line_index (GtkTextIter *iter,
                                      gint byte_on_line);
```

Like `gtk_text_iter_set_line_index()`, but the index is in visible bytes, i.e. text with a tag making it invisible is not counted in the index.

Parameters

iter	a GtkTextIter
byte_on_line	a byte index

---

gtk_text_iter_set_visible_line_offset ()

```
void
gtk_text_iter_set_visible_line_offset (GtkTextIter *iter,
                                       gint char_on_line);
```

Like `gtk_text_iter_set_line_offset()`, but the offset is in visible characters, i.e. text with a tag making it invisible is not counted in the offset.

Parameters

iter	a GtkTextIter
char_on_line	a character offset

---

gtk_text_iter_forward_to_end ()

```
void
gtk_text_iter_forward_to_end (GtkTextIter *iter);
```

Moves `iter` forward to the “end iterator,” which points one past the last valid character in the buffer. `gtk_text_iter_get_char()` called on the end iterator returns 0, which is convenient for writing loops.

Parameters

iter	a GtkTextIter

---

gtk_text_iter_forward_to_line_end ()

```
gboolean
gtk_text_iter_forward_to_line_end (GtkTextIter *iter);
```

Moves the iterator to point to the paragraph delimiter characters, which will be either a newline, a carriage return, a carriage return/newline in sequence, or the Unicode paragraph separator character. If the iterator is already at the paragraph delimiter characters, moves to the paragraph delimiter characters for the next line. If `iter` is on the last line in the buffer, which does not end in paragraph delimiters, moves to the end iterator (end of the last line), and returns `FALSE`.

Parameters

iter	a GtkTextIter

Returns

`TRUE` if we moved and the new location is not the end iterator

---

gtk_text_iter_forward_to_tag_toggle ()

```
gboolean
gtk_text_iter_forward_to_tag_toggle (GtkTextIter *iter,
                                     GtkTextTag *tag);
```

Moves forward to the next toggle (on or off) of the GtkTextTag `tag` , or to the next toggle of any tag if `tag` is `NULL`. If no matching tag toggles are found, returns `FALSE`, otherwise `TRUE`. Does not return toggles located at `iter` , only toggles after `iter` . Sets `iter` to the location of the toggle, or to the end of the buffer if no toggle is found.

Parameters

iter	a GtkTextIter	
tag	a GtkTextTag, or `NULL`.	[allow-none]

Returns

whether we found a tag toggle after `iter`

---

gtk_text_iter_backward_to_tag_toggle ()

```
gboolean
gtk_text_iter_backward_to_tag_toggle (GtkTextIter *iter,
                                      GtkTextTag *tag);
```

Moves backward to the next toggle (on or off) of the GtkTextTag `tag` , or to the next toggle of any tag if `tag` is `NULL`. If no matching tag toggles are found, returns `FALSE`, otherwise `TRUE`. Does not return toggles located at `iter` , only toggles before `iter` . Sets `iter` to the location of the toggle, or the start of the buffer if no toggle is found.

Parameters

iter	a GtkTextIter	
tag	a GtkTextTag, or `NULL`.	[allow-none]

Returns

whether we found a tag toggle before `iter`

GtkTextCharPredicate ()

```
gboolean
(*GtkTextCharPredicate) (gunichar ch,
                         gpointer user_data);
```

---

gtk_text_iter_forward_find_char ()

```
gboolean
gtk_text_iter_forward_find_char (GtkTextIter *iter,
                                 GtkTextCharPredicate pred,
                                 gpointer user_data,
                                 const GtkTextIter *limit);
```

Advances `iter` , calling `pred` on each character. If `pred` returns `TRUE`, returns `TRUE` and stops scanning. If `pred` never returns `TRUE`, `iter` is set to `limit` if `limit` is non-`NULL`, otherwise to the end iterator.

Parameters

iter	a GtkTextIter	
pred	a function to be called on each character.	[scope call]
user_data	user data for `pred`	
limit	search limit, or `NULL` for none.	[allow-none]

Returns

whether a match was found

---

gtk_text_iter_backward_find_char ()

```
gboolean
gtk_text_iter_backward_find_char (GtkTextIter *iter,
                                  GtkTextCharPredicate pred,
                                  gpointer user_data,
                                  const GtkTextIter *limit);
```

Same as `gtk_text_iter_forward_find_char()`, but goes backward from `iter` .

Parameters

iter	a GtkTextIter	
pred	function to be called on each character.	[scope call]
user_data	user data for `pred`	
limit	search limit, or `NULL` for none.	[allow-none]

Returns

whether a match was found

---

gtk_text_iter_forward_search ()

```
gboolean
gtk_text_iter_forward_search (const GtkTextIter *iter,
                              const gchar *str,
```

```
                                GtkTextSearchFlags flags,
                                GtkTextIter *match_start,
                                GtkTextIter *match_end,
                                const GtkTextIter *limit);
```

Searches forward for `str` . Any match is returned by setting `match_start` to the first character of the match and `match_end` to the first character after the match. The search will not continue past `limit` . Note that a search is a linear or O(n) operation, so you may wish to use `limit` to avoid locking up your UI on large buffers.

`match_start` will never be set to a GtkTextIter located before `iter` , even if there is a possible `match_end` after or at `iter` .

Parameters

iter	start of search	
str	a search string	
flags	flags affecting how the search is done	
match_start	return location for start of match, or `NULL`.	[out caller-allocates][allow-none]
match_end	return location for end of match, or `NULL`.	[out caller-allocates][allow-none]
limit	location of last possible `match_end` , or `NULL` for the end of the buffer.	[allow-none]

Returns

whether a match was found

---

gtk_text_iter_backward_search ()

```
gboolean
gtk_text_iter_backward_search (const GtkTextIter *iter,
                               const gchar *str,
                               GtkTextSearchFlags flags,
                               GtkTextIter *match_start,
                               GtkTextIter *match_end,
                               const GtkTextIter *limit);
```

Same as `gtk_text_iter_forward_search()`, but moves backward.

`match_end` will never be set to a GtkTextIter located after `iter` , even if there is a possible `match_start` before or at `iter` .

Parameters

iter	a GtkTextIter where the search begins	
str	search string	
flags	bitmask of flags affecting the search	
match_start	return location for start of match, or	[out caller-allocates][allow-none]

	NULL.	
match_end	return location for end of match, or NULL.	[out caller-allocates][allow-none]
limit	location of last possible match_start , or NULL for start of buffer.	[allow-none]

Returns

whether a match was found

---

gtk_text_iter_equal ()

```
gboolean
gtk_text_iter_equal (const GtkTextIter *lhs,
                     const GtkTextIter *rhs);
```

Tests whether two iterators are equal, using the fastest possible mechanism. This function is very fast; you can expect it to perform better than e.g. getting the character offset for each iterator and comparing the offsets yourself. Also, it's a bit faster than gtk_text_iter_compare().

Parameters

lhs	a GtkTextIter
rhs	another GtkTextIter

Returns

TRUE if the iterators point to the same place in the buffer

---

gtk_text_iter_compare ()

```
gint
gtk_text_iter_compare (const GtkTextIter *lhs,
                       const GtkTextIter *rhs);
```

A qsort()-style function that returns negative if lhs is less than rhs , positive if lhs is greater than rhs , and 0 if they're equal. Ordering is in character offset order, i.e. the first character in the buffer is less than the second character in the buffer.

Parameters

lhs	a GtkTextIter
rhs	another GtkTextIter

Returns

-1 if lhs is less than rhs , 1 if lhs is greater, 0 if they are equal

gtk_text_iter_in_range ()

```
gboolean
gtk_text_iter_in_range (const GtkTextIter *iter,
                        const GtkTextIter *start,
                        const GtkTextIter *end);
```

Checks whether `iter` falls in the range [`start` , `end`). `start` and `end` must be in ascending order.

Parameters

iter	a GtkTextIter
start	start of range
end	end of range

Returns

`TRUE` if `iter` is in the range

---

gtk_text_iter_order ()

```
void
gtk_text_iter_order (GtkTextIter *first,
                     GtkTextIter *second);
```

Swaps the value of `first` and `second` if `second` comes before `first` in the buffer. That is, ensures that `first` and `second` are in sequence. Most text buffer functions that take a range call this automatically on your behalf, so there's no real reason to call it yourself in those cases. There are some exceptions, such as `gtk_text_iter_in_range()`, that expect a pre-sorted range.

Parameters

first	a GtkTextIter
second	another GtkTextIter

Types and Values

GtkTextIter

```
typedef struct {
  /* GtkTextIter is an opaque datatype; ignore all these fields.
   * Initialize the iter with gtk_text_buffer_get_iter_*
   * functions
   */
} GtkTextIter;
```

---

enum GtkTextSearchFlags

Flags affecting how a search is done.

If neither GTK_TEXT_SEARCH_VISIBLE_ONLY nor GTK_TEXT_SEARCH_TEXT_ONLY are enabled, the match must be exact; the special 0xFFFC character will match embedded pixbufs or child widgets.

Members

GTK_TEXT_SEARCH_VISIBLE_ONLY	Search only visible data. A search match may have invisible text interspersed.
GTK_TEXT_SEARCH_TEXT_ONLY	Search only text. A match may have pixbufs or child widgets mixed inside the matched range.
GTK_TEXT_SEARCH_CASE_INSENSITIVE	The text will be matched regardless of what case it is in.

GtkTextMark

GtkTextMark — A position in the buffer preserved across buffer modifications

Functions

GtkTextMark *	gtk_text_mark_new ()
void	gtk_text_mark_set_visible ()
gboolean	gtk_text_mark_get_visible ()
gboolean	gtk_text_mark_get_deleted ()
const gchar *	gtk_text_mark_get_name ()
GtkTextBuffer *	gtk_text_mark_get_buffer ()
gboolean	gtk_text_mark_get_left_gravity ()

Properties

gboolean	left-gravity	Read / Write / Construct Only
gchar *	name	Read / Write / Construct Only

Types and Values

struct	GtkTextMark

Object Hierarchy

```
GObject
└── GtkTextMark
```

Includes

```
#include <gtk/gtk.h>
```

Description

You may wish to begin by reading the text widget conceptual overview which gives an overview of all the objects and data types related to the text widget and how they work together.

A GtkTextMark is like a bookmark in a text buffer; it preserves a position in the text. You can convert the mark to an iterator using `gtk_text_buffer_get_iter_at_mark()`. Unlike iterators, marks remain valid across buffer mutations, because their behavior is defined when text is inserted or deleted. When text containing a mark is deleted, the mark remains in the position originally occupied by the deleted text. When text is inserted at a mark, a mark with “left gravity” will be moved to the beginning of the newly-inserted text, and a mark with “right gravity” will be moved to the end.

Note that “left” and “right” here refer to logical direction (left is the toward the start of the buffer); in some languages such as Hebrew the logically-leftmost text is not actually on the left when displayed.

Marks are reference counted, but the reference count only controls the validity of the memory; marks can be deleted from the buffer at any time with `gtk_text_buffer_delete_mark()`. Once deleted from the buffer, a mark is essentially useless.

Marks optionally have names; these can be convenient to avoid passing the GtkTextMark object around.

Marks are typically created using the `gtk_text_buffer_create_mark()` function.

Functions

gtk_text_mark_new ()

```
GtkTextMark *
gtk_text_mark_new (const gchar *name,
                   gboolean left_gravity);
```

Creates a text mark. Add it to a buffer using `gtk_text_buffer_add_mark()`. If `name` is `NULL`, the mark is anonymous; otherwise, the mark can be retrieved by name using `gtk_text_buffer_get_mark()`. If a mark has left gravity, and text is inserted at the mark’s current location, the mark will be moved to the left of the newly-inserted text. If the mark has right gravity (`left_gravity` = `FALSE`), the mark will end up on the right of newly-inserted text. The standard left-to-right cursor is a mark with right gravity (when you type, the cursor stays on the right side of the text you’re typing).

Parameters

name	mark name or `NULL`.	[allow-none]
left_gravity	whether the mark should have left gravity	

Returns

new GtkTextMark

Since: 2.12

---

gtk_text_mark_set_visible ()

```
void
gtk_text_mark_set_visible (GtkTextMark *mark,
                           gboolean setting);
```

Sets the visibility of `mark` ; the insertion point is normally visible, i.e. you can see it as a vertical bar. Also, the text widget uses a visible mark to indicate where a drop will occur when dragging-and-dropping text. Most other marks are not visible. Marks are not visible by default.

Parameters

mark	a GtkTextMark
setting	visibility of mark

---

gtk_text_mark_get_visible ()

```
gboolean
gtk_text_mark_get_visible (GtkTextMark *mark);
```

Returns `TRUE` if the mark is visible (i.e. a cursor is displayed for it).

Parameters

mark	a GtkTextMark

Returns

`TRUE` if visible

---

gtk_text_mark_get_deleted ()

```
gboolean
gtk_text_mark_get_deleted (GtkTextMark *mark);
```

Returns `TRUE` if the mark has been removed from its buffer with `gtk_text_buffer_delete_mark()`. See `gtk_text_buffer_add_mark()` for a way to add it to a buffer again.

Parameters

mark	a GtkTextMark

Returns

whether the mark is deleted

---

gtk_text_mark_get_name ()

```
const gchar *
gtk_text_mark_get_name (GtkTextMark *mark);
```

Returns the mark name; returns NULL for anonymous marks.

Parameters

mark	a GtkTextMark

Returns

mark name.

[nullable]

---

gtk_text_mark_get_buffer ()

```
GtkTextBuffer *
gtk_text_mark_get_buffer (GtkTextMark *mark);
```

Gets the buffer this mark is located inside, or `NULL` if the mark is deleted.

Parameters

mark	a GtkTextMark

Returns

the mark's GtkTextBuffer.

[transfer none]

---

gtk_text_mark_get_left_gravity ()

```
gboolean
gtk_text_mark_get_left_gravity (GtkTextMark *mark);
```

Determines whether the mark has left gravity.

Parameters

mark	a GtkTextMark

Returns

`TRUE` if the mark has left gravity, `FALSE` otherwise

Types and Values

struct GtkTextMark

```
struct GtkTextMark;
```

Property Details

The “left-gravity” property

```
  “left-gravity”             gboolean
```

Whether the mark has left gravity. When text is inserted at the mark’s current location, if the mark has left gravity it will be moved to the left of the newly-inserted text, otherwise to the right.

Flags: Read / Write / Construct Only

Default value: FALSE

The “name” property

```
  “name”                     gchar *
```

The name of the mark or `NULL` if the mark is anonymous.

Flags: Read / Write / Construct Only

Default value: NULL

GtkTextBuffer

GtkTextBuffer — Stores attributed text for display in a GtkTextView

Functions

GtkTextBuffer *	gtk_text_buffer_new ()
gint	gtk_text_buffer_get_line_count ()
gint	gtk_text_buffer_get_char_count ()

GtkTextTagTable *	gtk_text_buffer_get_tag_table ()
void	gtk_text_buffer_insert ()
void	gtk_text_buffer_insert_at_cursor ()
gboolean	gtk_text_buffer_insert_interactive ()
gboolean	gtk_text_buffer_insert_interactive_at_cursor ()
void	gtk_text_buffer_insert_range ()
gboolean	gtk_text_buffer_insert_range_interactive ()
void	gtk_text_buffer_insert_with_tags ()
void	gtk_text_buffer_insert_with_tags_by_name ()
void	gtk_text_buffer_insert_markup ()
void	gtk_text_buffer_delete ()
gboolean	gtk_text_buffer_delete_interactive ()
gboolean	gtk_text_buffer_backspace ()
void	gtk_text_buffer_set_text ()
gchar *	gtk_text_buffer_get_text ()
gchar *	gtk_text_buffer_get_slice ()
void	gtk_text_buffer_insert_pixbuf ()
void	gtk_text_buffer_insert_child_anchor ()
GtkTextChildAnchor *	gtk_text_buffer_create_child_anchor ()
GtkTextMark *	gtk_text_buffer_create_mark ()
void	gtk_text_buffer_move_mark ()
void	gtk_text_buffer_move_mark_by_name ()
void	gtk_text_buffer_add_mark ()
void	gtk_text_buffer_delete_mark ()
void	gtk_text_buffer_delete_mark_by_name ()
GtkTextMark *	gtk_text_buffer_get_mark ()
GtkTextMark *	gtk_text_buffer_get_insert ()
GtkTextMark *	gtk_text_buffer_get_selection_bound ()
gboolean	gtk_text_buffer_get_has_selection ()
void	gtk_text_buffer_place_cursor ()
void	gtk_text_buffer_select_range ()
void	gtk_text_buffer_apply_tag ()
void	gtk_text_buffer_remove_tag ()
void	gtk_text_buffer_apply_tag_by_name ()
void	gtk_text_buffer_remove_tag_by_name ()
void	gtk_text_buffer_remove_all_tags ()
GtkTextTag *	gtk_text_buffer_create_tag ()
void	gtk_text_buffer_get_iter_at_line_offset ()
void	gtk_text_buffer_get_iter_at_offset ()
void	gtk_text_buffer_get_iter_at_line ()
void	gtk_text_buffer_get_iter_at_line_index ()
void	gtk_text_buffer_get_iter_at_mark ()
void	gtk_text_buffer_get_iter_at_child_anchor ()
void	gtk_text_buffer_get_start_iter ()
void	gtk_text_buffer_get_end_iter ()
void	gtk_text_buffer_get_bounds ()
gboolean	gtk_text_buffer_get_modified ()
void	gtk_text_buffer_set_modified ()
gboolean	gtk_text_buffer_delete_selection ()
void	gtk_text_buffer_paste_clipboard ()
void	gtk_text_buffer_copy_clipboard ()
void	gtk_text_buffer_cut_clipboard ()

gboolean	gtk_text_buffer_get_selection_bounds ()
void	gtk_text_buffer_begin_user_action ()
void	gtk_text_buffer_end_user_action ()
void	gtk_text_buffer_add_selection_clipboard ()
void	gtk_text_buffer_remove_selection_clipboard ()
gboolean	(*GtkTextBufferDeserializeFunc) ()
gboolean	gtk_text_buffer_deserialize ()
gboolean	gtk_text_buffer_deserialize_get_can_create_tags ()
void	gtk_text_buffer_deserialize_set_can_create_tags ()
GtkTargetList *	gtk_text_buffer_get_copy_target_list ()
GdkAtom *	gtk_text_buffer_get_deserialize_formats ()
GtkTargetList *	gtk_text_buffer_get_paste_target_list ()
GdkAtom *	gtk_text_buffer_get_serialize_formats ()
GdkAtom	gtk_text_buffer_register_deserialize_format ()
GdkAtom	gtk_text_buffer_register_deserialize_tagset ()
GdkAtom	gtk_text_buffer_register_serialize_format ()
GdkAtom	gtk_text_buffer_register_serialize_tagset ()
guint8 *	(*GtkTextBufferSerializeFunc) ()
guint8 *	gtk_text_buffer_serialize ()
void	gtk_text_buffer_unregister_deserialize_format ()
void	gtk_text_buffer_unregister_serialize_format ()

Properties

GtkTargetList *	copy-target-list	Read
gint	cursor-position	Read
gboolean	has-selection	Read
GtkTargetList *	paste-target-list	Read
GtkTextTagTable *	tag-table	Read / Write / Construct Only
gchar *	text	Read / Write

Signals

void	apply-tag	Run Last
void	begin-user-action	Run Last
void	changed	Run Last
void	delete-range	Run Last
void	end-user-action	Run Last
void	insert-child-anchor	Run Last
void	insert-pixbuf	Run Last
void	insert-text	Run Last
void	mark-deleted	Run Last
void	mark-set	Run Last
void	modified-changed	Run Last
void	paste-done	Run Last
void	remove-tag	Run Last

Types and Values

GtkTextBuffer

struct	GtkTextBufferClass
enum	GtkTextBufferTargetInfo

Object Hierarchy

```
GObject
└── GtkTextBuffer
```

Includes

```
#include <gtk/gtk.h>
```

Description

You may wish to begin by reading the text widget conceptual overview which gives an overview of all the objects and data types related to the text widget and how they work together.

Functions

gtk_text_buffer_new ()

```
GtkTextBuffer *
gtk_text_buffer_new (GtkTextTagTable *table);
```

Creates a new text buffer.

Parameters

table	a tag table, or `NULL` to create a new one.	[allow-none]

Returns

a new text buffer

---

gtk_text_buffer_get_line_count ()

```
gint
gtk_text_buffer_get_line_count (GtkTextBuffer *buffer);
```

Obtains the number of lines in the buffer. This value is cached, so the function is very fast.

Parameters

buffer	a GtkTextBuffer

Returns

number of lines in the buffer

---

gtk_text_buffer_get_char_count ()

```
gint
gtk_text_buffer_get_char_count (GtkTextBuffer *buffer);
```

Gets the number of characters in the buffer; note that characters and bytes are not the same, you can't e.g. expect the contents of the buffer in string form to be this many bytes long. The character count is cached, so this function is very fast.

Parameters

buffer	a GtkTextBuffer

Returns

number of characters in the buffer

---

gtk_text_buffer_get_tag_table ()

```
GtkTextTagTable *
gtk_text_buffer_get_tag_table (GtkTextBuffer *buffer);
```

Get the GtkTextTagTable associated with this buffer.

Parameters

buffer	a GtkTextBuffer

Returns

the buffer's tag table.

[transfer none]

---

gtk_text_buffer_insert ()

```
void
gtk_text_buffer_insert (GtkTextBuffer *buffer,
                        GtkTextIter *iter,
                        const gchar *text,
                        gint len);
```

Inserts `len` bytes of `text` at position `iter` . If `len` is -1, `text` must be nul-terminated and will be inserted in its entirety. Emits the "insert-text" signal; insertion actually occurs in the default handler for the signal. `iter` is invalidated when insertion occurs (because the buffer contents change), but the default signal handler

revalidates it to point to the end of the inserted text.

Parameters

buffer	a GtkTextBuffer
iter	a position in the buffer
text	text in UTF-8 format
len	length of text in bytes, or -1

---

gtk_text_buffer_insert_at_cursor ()

```
void
gtk_text_buffer_insert_at_cursor (GtkTextBuffer *buffer,
                                  const gchar *text,
                                  gint len);
```

Simply calls `gtk_text_buffer_insert()`, using the current cursor position as the insertion point.

Parameters

buffer	a GtkTextBuffer
text	text in UTF-8 format
len	length of text, in bytes

---

gtk_text_buffer_insert_interactive ()

```
gboolean
gtk_text_buffer_insert_interactive (GtkTextBuffer *buffer,
                                    GtkTextIter *iter,
                                    const gchar *text,
                                    gint len,
                                    gboolean default_editable);
```

Like `gtk_text_buffer_insert()`, but the insertion will not occur if `iter` is at a non-editable location in the buffer. Usually you want to prevent insertions at ineditable locations if the insertion results from a user action (is interactive).

`default_editable` indicates the editability of text that doesn't have a tag affecting editability applied to it. Typically the result of `gtk_text_view_get_editable()` is appropriate here.

Parameters

buffer	a GtkTextBuffer
iter	a position in `buffer`
text	some UTF-8 text
len	length of text in bytes, or -1
default_editable	default editability of buffer

Returns

whether text was actually inserted

---

gtk_text_buffer_insert_interactive_at_cursor ()

```
gboolean
gtk_text_buffer_insert_interactive_at_cursor
                                (GtkTextBuffer *buffer,
                                 const gchar *text,
                                 gint len,
                                 gboolean default_editable);
```

Calls `gtk_text_buffer_insert_interactive()` at the cursor position.

`default_editable` indicates the editability of text that doesn't have a tag affecting editability applied to it. Typically the result of `gtk_text_view_get_editable()` is appropriate here.

Parameters

buffer	a GtkTextBuffer
text	text in UTF-8 format
len	length of text in bytes, or -1
default_editable	default editability of buffer

Returns

whether text was actually inserted

---

gtk_text_buffer_insert_range ()

```
void
gtk_text_buffer_insert_range (GtkTextBuffer *buffer,
                              GtkTextIter *iter,
                              const GtkTextIter *start,
                              const GtkTextIter *end);
```

Copies text, tags, and pixbufs between `start` and `end` (the order of `start` and `end` doesn’t matter) and inserts the copy at `iter` . Used instead of simply getting/inserting text because it preserves images and tags. If `start` and `end` are in a different buffer from `buffer` , the two buffers must share the same tag table.

Implemented via emissions of the insert_text and apply_tag signals, so expect those.

Parameters

buffer	a GtkTextBuffer
iter	a position in `buffer`
start	a position in a GtkTextBuffer
end	another position in the same buffer as `start`

---

gtk_text_buffer_insert_range_interactive ()

```
gboolean
gtk_text_buffer_insert_range_interactive
                                (GtkTextBuffer *buffer,
                                 GtkTextIter *iter,
                                 const GtkTextIter *start,
                                 const GtkTextIter *end,
                                 gboolean default_editable);
```

Same as `gtk_text_buffer_insert_range()`, but does nothing if the insertion point isn't editable. The `default_editable` parameter indicates whether the text is editable at `iter` if no tags enclosing `iter` affect editability. Typically the result of `gtk_text_view_get_editable()` is appropriate here.

Parameters

buffer	a GtkTextBuffer
iter	a position in `buffer`
start	a position in a GtkTextBuffer
end	another position in the same buffer as `start`
default_editable	default editability of the buffer

Returns

whether an insertion was possible at `iter`

---

gtk_text_buffer_insert_with_tags ()

```
void
gtk_text_buffer_insert_with_tags (GtkTextBuffer *buffer,
                                  GtkTextIter *iter,
                                  const gchar *text,
                                  gint len,
                                  GtkTextTag *first_tag,
                                  ...);
```

Inserts `text` into `buffer` at `iter` , applying the list of tags to the newly-inserted text. The last tag specified must be `NULL` to terminate the list. Equivalent to calling `gtk_text_buffer_insert()`, then `gtk_text_buffer_apply_tag()` on the inserted text; `gtk_text_buffer_insert_with_tags()` is just a convenience function.

Parameters

buffer	a GtkTextBuffer
iter	an iterator in `buffer`
text	UTF-8 text
len	length of `text` , or -1
first_tag	first tag to apply to `text`
...	`NULL`-terminated list of tags to apply

---

gtk_text_buffer_insert_with_tags_by_name ()

```
void
gtk_text_buffer_insert_with_tags_by_name
                                (GtkTextBuffer *buffer,
                                 GtkTextIter *iter,
                                 const gchar *text,
                                 gint len,
                                 const gchar *first_tag_name,
                                 ...);
```

Same as `gtk_text_buffer_insert_with_tags()`, but allows you to pass in tag names instead of tag objects.

Parameters

buffer	a GtkTextBuffer
iter	position in `buffer`
text	UTF-8 text
len	length of `text` , or -1
first_tag_name	name of a tag to apply to `text`
...	more tag names

---

gtk_text_buffer_insert_markup ()

```
void
gtk_text_buffer_insert_markup (GtkTextBuffer *buffer,
                               GtkTextIter *iter,
                               const gchar *markup,
                               gint len);
```

Inserts the text in `markup` at position `iter` . `markup` will be inserted in its entirety and must be nul-terminated and valid UTF-8. Emits the "insert-text" signal, possibly multiple times; insertion actually occurs in the default handler for the signal. `iter` will point to the end of the inserted text on return.

Parameters

buffer	a GtkTextBuffer
iter	location to insert the markup
markup	a nul-terminated UTF-8 string containing Pango markup
len	length of `markup` in bytes, or -1

Since: 3.16

---

gtk_text_buffer_delete ()

```
void
gtk_text_buffer_delete (GtkTextBuffer *buffer,
                        GtkTextIter *start,
                        GtkTextIter *end);
```

Deletes text between `start` and `end` . The order of `start` and `end` is not actually relevant; `gtk_text_buffer_delete()` will reorder them. This function actually emits the “delete-range” signal, and the default handler of that signal deletes the text. Because the buffer is modified, all outstanding iterators become invalid after calling this function; however, the `start` and `end` will be re-initialized to point to the location where text was deleted.

Parameters

buffer	a GtkTextBuffer
start	a position in `buffer`
end	another position in `buffer`

---

gtk_text_buffer_delete_interactive ()

```
gboolean
gtk_text_buffer_delete_interactive (GtkTextBuffer *buffer,
                                    GtkTextIter *start_iter,
                                    GtkTextIter *end_iter,
                                    gboolean default_editable);
```

Deletes all editable text in the given range. Calls `gtk_text_buffer_delete()` for each editable sub-range of [`start` ,`end`). `start` and `end` are revalidated to point to the location of the last deleted range, or left untouched if no text was deleted.

Parameters

buffer	a GtkTextBuffer
start_iter	start of range to delete
end_iter	end of range
default_editable	whether the buffer is editable by default

Returns

whether some text was actually deleted

---

gtk_text_buffer_backspace ()

```
gboolean
gtk_text_buffer_backspace (GtkTextBuffer *buffer,
                           GtkTextIter *iter,
                           gboolean interactive,
                           gboolean default_editable);
```

Performs the appropriate action as if the user hit the delete key with the cursor at the position specified by `iter` . In the normal case a single character will be deleted, but when combining accents are involved, more than one character can be deleted, and when precomposed character and accent combinations are involved, less than one character will be deleted.

Because the buffer is modified, all outstanding iterators become invalid after calling this function; however, the

iter will be re-initialized to point to the location where text was deleted.

Parameters

buffer	a GtkTextBuffer
iter	a position in buffer
interactive	whether the deletion is caused by user interaction
default_editable	whether the buffer is editable by default

Returns

TRUE if the buffer was modified

Since: 2.6

---

gtk_text_buffer_set_text ()

```
void
gtk_text_buffer_set_text (GtkTextBuffer *buffer,
                          const gchar *text,
                          gint len);
```

Deletes current contents of buffer , and inserts text instead. If len is -1, text must be nul-terminated. text must be valid UTF-8.

Parameters

buffer	a GtkTextBuffer
text	UTF-8 text to insert
len	length of text in bytes

---

gtk_text_buffer_get_text ()

```
gchar *
gtk_text_buffer_get_text (GtkTextBuffer *buffer,
                          const GtkTextIter *start,
                          const GtkTextIter *end,
                          gboolean include_hidden_chars);
```

Returns the text in the range [start ,end). Excludes undisplayed text (text marked with tags that set the invisibility attribute) if include_hidden_chars is FALSE. Does not include characters representing embedded images, so byte and character indexes into the returned string do not correspond to byte and character indexes into the buffer. Contrast with gtk_text_buffer_get_slice().

Parameters

buffer	a GtkTextBuffer
start	start of a range
end	end of a range
include_hidden_chars	whether to include invisible text

Returns

an allocated UTF-8 string.

[transfer full]

---

gtk_text_buffer_get_slice ()

```
gchar *
gtk_text_buffer_get_slice (GtkTextBuffer *buffer,
                           const GtkTextIter *start,
                           const GtkTextIter *end,
                           gboolean include_hidden_chars);
```

Returns the text in the range [`start` ,`end`). Excludes undisplayed text (text marked with tags that set the invisibility attribute) if `include_hidden_chars` is `FALSE`. The returned string includes a 0xFFFC character whenever the buffer contains embedded images, so byte and character indexes into the returned string do correspond to byte and character indexes into the buffer. Contrast with `gtk_text_buffer_get_text()`. Note that 0xFFFC can occur in normal text as well, so it is not a reliable indicator that a pixbuf or widget is in the buffer.

Parameters

buffer	a GtkTextBuffer
start	start of a range
end	end of a range
include_hidden_chars	whether to include invisible text

Returns

an allocated UTF-8 string.

[transfer full]

---

gtk_text_buffer_insert_pixbuf ()

```
void
gtk_text_buffer_insert_pixbuf (GtkTextBuffer *buffer,
                               GtkTextIter *iter,
                               GdkPixbuf *pixbuf);
```

Inserts an image into the text buffer at `iter` . The image will be counted as one character in character counts, and when obtaining the buffer contents as a string, will be represented by the Unicode “object replacement character” 0xFFFC. Note that the “slice” variants for obtaining portions of the buffer as a string include this

character for pixbufs, but the “text” variants do not. e.g. see gtk_text_buffer_get_slice() and gtk_text_buffer_get_text().

Parameters

buffer	a GtkTextBuffer
iter	location to insert the pixbuf
pixbuf	a GdkPixbuf

---

gtk_text_buffer_insert_child_anchor ()

```
void
gtk_text_buffer_insert_child_anchor (GtkTextBuffer *buffer,
                                     GtkTextIter *iter,
                                     GtkTextChildAnchor *anchor);
```

Inserts a child widget anchor into the text buffer at `iter` . The anchor will be counted as one character in character counts, and when obtaining the buffer contents as a string, will be represented by the Unicode “object replacement character” 0xFFFC. Note that the “slice” variants for obtaining portions of the buffer as a string include this character for child anchors, but the “text” variants do not. E.g. see gtk_text_buffer_get_slice() and gtk_text_buffer_get_text(). Consider gtk_text_buffer_create_child_anchor() as a more convenient alternative to this function. The buffer will add a reference to the anchor, so you can unref it after insertion.

Parameters

buffer	a GtkTextBuffer
iter	location to insert the anchor
anchor	a GtkTextChildAnchor

---

gtk_text_buffer_create_child_anchor ()

```
GtkTextChildAnchor *
gtk_text_buffer_create_child_anchor (GtkTextBuffer *buffer,
                                     GtkTextIter *iter);
```

This is a convenience function which simply creates a child anchor with gtk_text_child_anchor_new() and inserts it into the buffer with gtk_text_buffer_insert_child_anchor(). The new anchor is owned by the buffer; no reference count is returned to the caller of gtk_text_buffer_create_child_anchor().

Parameters

buffer	a GtkTextBuffer
iter	location in the buffer

Returns

the created child anchor.

[transfer none]

---

gtk_text_buffer_create_mark ()

```
GtkTextMark *
gtk_text_buffer_create_mark (GtkTextBuffer *buffer,
                             const gchar *mark_name,
                             const GtkTextIter *where,
                             gboolean left_gravity);
```

Creates a mark at position `where` . If `mark_name` is `NULL`, the mark is anonymous; otherwise, the mark can be retrieved by name using `gtk_text_buffer_get_mark()`. If a mark has left gravity, and text is inserted at the mark's current location, the mark will be moved to the left of the newly-inserted text. If the mark has right gravity (`left_gravity` = `FALSE`), the mark will end up on the right of newly-inserted text. The standard left-to-right cursor is a mark with right gravity (when you type, the cursor stays on the right side of the text you're typing).

The caller of this function does not own a reference to the returned GtkTextMark, so you can ignore the return value if you like. Marks are owned by the buffer and go away when the buffer does.

Emits the "mark-set" signal as notification of the mark's initial placement.

Parameters

buffer	a GtkTextBuffer	
mark_name	name for mark, or `NULL`.	[allow-none]
where	location to place mark	
left_gravity	whether the mark has left gravity	

Returns

the new GtkTextMark object.

[transfer none]

---

gtk_text_buffer_move_mark ()

```
void
gtk_text_buffer_move_mark (GtkTextBuffer *buffer,
                           GtkTextMark *mark,
                           const GtkTextIter *where);
```

Moves `mark` to the new location `where` . Emits the "mark-set" signal as notification of the move.

Parameters

buffer	a GtkTextBuffer

mark	a GtkTextMark
where	new location for `mark` in `buffer`

---

gtk_text_buffer_move_mark_by_name ()

```
void
gtk_text_buffer_move_mark_by_name (GtkTextBuffer *buffer,
                                   const gchar *name,
                                   const GtkTextIter *where);
```

Moves the mark named `name` (which must exist) to location `where` . See `gtk_text_buffer_move_mark()` for details.

Parameters

buffer	a GtkTextBuffer
name	name of a mark
where	new location for mark

---

gtk_text_buffer_add_mark ()

```
void
gtk_text_buffer_add_mark (GtkTextBuffer *buffer,
                          GtkTextMark *mark,
                          const GtkTextIter *where);
```

Adds the mark at position `where` . The mark must not be added to another buffer, and if its name is not `NULL` then there must not be another mark in the buffer with the same name.

Emits the "mark-set" signal as notification of the mark's initial placement.

Parameters

buffer	a GtkTextBuffer
mark	the mark to add
where	location to place mark

Since: 2.12

---

gtk_text_buffer_delete_mark ()

```
void
gtk_text_buffer_delete_mark (GtkTextBuffer *buffer,
                             GtkTextMark *mark);
```

Deletes `mark` , so that it's no longer located anywhere in the buffer. Removes the reference the buffer holds to the mark, so if you haven't called `g_object_ref()` on the mark, it will be freed. Even if the mark isn't freed, most operations on `mark` become invalid, until it gets added to a buffer again with `gtk_text_buffer_add_mark()`. Use `gtk_text_mark_get_deleted()` to find out if a mark has been removed from its buffer. The "mark-deleted" signal will be emitted as notification after the mark is deleted.

Parameters

buffer	a GtkTextBuffer
mark	a GtkTextMark in `buffer`

---

gtk_text_buffer_delete_mark_by_name ()

```
void
gtk_text_buffer_delete_mark_by_name (GtkTextBuffer *buffer,
                                     const gchar *name);
```

Deletes the mark named `name` ; the mark must exist. See `gtk_text_buffer_delete_mark()` for details.

Parameters

buffer	a GtkTextBuffer
name	name of a mark in `buffer`

---

gtk_text_buffer_get_mark ()

```
GtkTextMark *
gtk_text_buffer_get_mark (GtkTextBuffer *buffer,
                          const gchar *name);
```

Returns the mark named `name` in buffer `buffer` , or `NULL` if no such mark exists in the buffer.

Parameters

buffer	a GtkTextBuffer
name	a mark name

Returns

a GtkTextMark, or `NULL`.

[nullable][transfer none]

---

gtk_text_buffer_get_insert ()

```
GtkTextMark *
gtk_text_buffer_get_insert (GtkTextBuffer *buffer);
```

Returns the mark that represents the cursor (insertion point). Equivalent to calling `gtk_text_buffer_get_mark()` to get the mark named “insert”, but very slightly more efficient, and involves less typing.

Parameters

buffer	a GtkTextBuffer

Returns

insertion point mark.

[transfer none]

---

gtk_text_buffer_get_selection_bound ()

```
GtkTextMark *
gtk_text_buffer_get_selection_bound (GtkTextBuffer *buffer);
```

Returns the mark that represents the selection bound. Equivalent to calling `gtk_text_buffer_get_mark()` to get the mark named “selection_bound”, but very slightly more efficient, and involves less typing.

The currently-selected text in `buffer` is the region between the “selection_bound” and “insert” marks. If “selection_bound” and “insert” are in the same place, then there is no current selection. `gtk_text_buffer_get_selection_bounds()` is another convenient function for handling the selection, if you just want to know whether there’s a selection and what its bounds are.

Parameters

buffer	a GtkTextBuffer

Returns

selection bound mark.

[transfer none]

---

gtk_text_buffer_get_has_selection ()

```
gboolean
gtk_text_buffer_get_has_selection (GtkTextBuffer *buffer);
```

Indicates whether the buffer has some text currently selected.

Parameters

buffer	a GtkTextBuffer

Returns

`TRUE` if the there is text selected

Since: 2.10

---

gtk_text_buffer_place_cursor ()

```
void
gtk_text_buffer_place_cursor (GtkTextBuffer *buffer,
                              const GtkTextIter *where);
```

This function moves the “insert” and “selection_bound” marks simultaneously. If you move them to the same place in two steps with `gtk_text_buffer_move_mark()`, you will temporarily select a region in between their old and new locations, which can be pretty inefficient since the temporarily-selected region will force stuff to be recalculated. This function moves them as a unit, which can be optimized.

Parameters

buffer	a GtkTextBuffer
where	where to put the cursor

---

gtk_text_buffer_select_range ()

```
void
gtk_text_buffer_select_range (GtkTextBuffer *buffer,
                              const GtkTextIter *ins,
                              const GtkTextIter *bound);
```

This function moves the “insert” and “selection_bound” marks simultaneously. If you move them in two steps with `gtk_text_buffer_move_mark()`, you will temporarily select a region in between their old and new locations, which can be pretty inefficient since the temporarily-selected region will force stuff to be recalculated. This function moves them as a unit, which can be optimized.

Parameters

buffer	a GtkTextBuffer
ins	where to put the “insert” mark
bound	where to put the “selection_bound” mark

Since: 2.4

---

gtk_text_buffer_apply_tag ()

```
void
gtk_text_buffer_apply_tag (GtkTextBuffer *buffer,
                           GtkTextTag *tag,
                           const GtkTextIter *start,
                           const GtkTextIter *end);
```

Emits the “apply-tag” signal on `buffer` . The default handler for the signal applies `tag` to the given range. `start` and `end` do not have to be in order.

Parameters

buffer	a GtkTextBuffer
tag	a GtkTextTag
start	one bound of range to be tagged
end	other bound of range to be tagged

---

gtk_text_buffer_remove_tag ()

```
void
gtk_text_buffer_remove_tag (GtkTextBuffer *buffer,
                            GtkTextTag *tag,
                            const GtkTextIter *start,
                            const GtkTextIter *end);
```

Emits the “remove-tag” signal. The default handler for the signal removes all occurrences of `tag` from the given range. `start` and `end` don’t have to be in order.

Parameters

buffer	a GtkTextBuffer
tag	a GtkTextTag
start	one bound of range to be untagged
end	other bound of range to be untagged

---

gtk_text_buffer_apply_tag_by_name ()

```
void
gtk_text_buffer_apply_tag_by_name (GtkTextBuffer *buffer,
                                   const gchar *name,
                                   const GtkTextIter *start,
                                   const GtkTextIter *end);
```

Calls `gtk_text_tag_table_lookup()` on the buffer’s tag table to get a GtkTextTag, then calls `gtk_text_buffer_apply_tag()`.

Parameters

buffer	a GtkTextBuffer
name	name of a named GtkTextTag
start	one bound of range to be tagged
end	other bound of range to be tagged

---

gtk_text_buffer_remove_tag_by_name ()

```
void
gtk_text_buffer_remove_tag_by_name (GtkTextBuffer *buffer,
                                    const gchar *name,
                                    const GtkTextIter *start,
```

```
                                const GtkTextIter *end);
```

Calls gtk_text_tag_table_lookup() on the buffer's tag table to get a GtkTextTag, then calls gtk_text_buffer_remove_tag().

Parameters

buffer	a GtkTextBuffer
name	name of a GtkTextTag
start	one bound of range to be untagged
end	other bound of range to be untagged

---

gtk_text_buffer_remove_all_tags ()

```
void
gtk_text_buffer_remove_all_tags (GtkTextBuffer *buffer,
                                 const GtkTextIter *start,
                                 const GtkTextIter *end);
```

Removes all tags in the range between `start` and `end` . Be careful with this function; it could remove tags added in code unrelated to the code you're currently writing. That is, using this function is probably a bad idea if you have two or more unrelated code sections that add tags.

Parameters

buffer	a GtkTextBuffer
start	one bound of range to be untagged
end	other bound of range to be untagged

---

gtk_text_buffer_create_tag ()

```
GtkTextTag *
gtk_text_buffer_create_tag (GtkTextBuffer *buffer,
                            const gchar *tag_name,
                            const gchar *first_property_name,
                            ...);
```

Creates a tag and adds it to the tag table for `buffer` . Equivalent to calling gtk_text_tag_new() and then adding the tag to the buffer's tag table. The returned tag is owned by the buffer's tag table, so the ref count will be equal to one.

If `tag_name` is `NULL`, the tag is anonymous.

If `tag_name` is non-`NULL`, a tag called `tag_name` must not already exist in the tag table for this buffer.

The `first_property_name` argument and subsequent arguments are a list of properties to set on the tag, as with `g_object_set()`.

Parameters

buffer	a GtkTextBuffer	
tag_name	name of the new tag, or `NULL`.	[allow-none]
first_property_name	name of first property to set, or `NULL`.	[allow-none]
...	`NULL`-terminated list of property names and values	

Returns

a new tag.

[transfer none]

---

gtk_text_buffer_get_iter_at_line_offset ()

```
void
gtk_text_buffer_get_iter_at_line_offset
                                (GtkTextBuffer *buffer,
                                 GtkTextIter *iter,
                                 gint line_number,
                                 gint char_offset);
```

Obtains an iterator pointing to `char_offset` within the given line. Note characters, not bytes; UTF-8 may encode one character as multiple bytes.

Before the 3.20 version, it was not allowed to pass an invalid location.

Since the 3.20 version, if `line_number` is greater than the number of lines in the `buffer` , the end iterator is returned. And if `char_offset` is off the end of the line, the iterator at the end of the line is returned.

Parameters

buffer	a GtkTextBuffer	
iter	iterator to initialize.	[out]
line_number	line number counting from 0	
char_offset	char offset from start of line	

---

gtk_text_buffer_get_iter_at_offset ()

```
void
gtk_text_buffer_get_iter_at_offset (GtkTextBuffer *buffer,
                                    GtkTextIter *iter,
                                    gint char_offset);
```

Initializes `iter` to a position `char_offset` chars from the start of the entire buffer. If `char_offset` is -1 or greater than the number of characters in the buffer, `iter` is initialized to the end iterator, the iterator one past the last valid character in the buffer.

Parameters

buffer	a GtkTextBuffer	
iter	iterator to initialize.	[out]
char_offset	char offset from start of buffer, counting from 0, or -1	

---

gtk_text_buffer_get_iter_at_line ()

```
void
gtk_text_buffer_get_iter_at_line (GtkTextBuffer *buffer,
                                  GtkTextIter *iter,
                                  gint line_number);
```

Initializes `iter` to the start of the given line. If `line_number` is greater than the number of lines in the `buffer` , the end iterator is returned.

Parameters

buffer	a GtkTextBuffer	
iter	iterator to initialize.	[out]
line_number	line number counting from 0	

---

gtk_text_buffer_get_iter_at_line_index ()

```
void
gtk_text_buffer_get_iter_at_line_index
                                (GtkTextBuffer *buffer,
                                 GtkTextIter *iter,
                                 gint line_number,
                                 gint byte_index);
```

Obtains an iterator pointing to `byte_index` within the given line. `byte_index` must be the start of a UTF-8 character. Note bytes, not characters; UTF-8 may encode one character as multiple bytes.

Before the 3.20 version, it was not allowed to pass an invalid location.

Since the 3.20 version, if `line_number` is greater than the number of lines in the `buffer` , the end iterator is returned. And if `byte_index` is off the end of the line, the iterator at the end of the line is returned.

Parameters

buffer	a GtkTextBuffer	
iter	iterator to initialize.	[out]
line_number	line number counting from 0	
byte_index	byte index from start of line	

---

gtk_text_buffer_get_iter_at_mark ()

```
void
gtk_text_buffer_get_iter_at_mark (GtkTextBuffer *buffer,
                                  GtkTextIter *iter,
                                  GtkTextMark *mark);
```

Initializes `iter` with the current position of `mark` .

Parameters

buffer	a GtkTextBuffer	
iter	iterator to initialize.	[out]
mark	a GtkTextMark in `buffer`	

---

gtk_text_buffer_get_iter_at_child_anchor ()

```
void
gtk_text_buffer_get_iter_at_child_anchor
                                 (GtkTextBuffer *buffer,
                                  GtkTextIter *iter,
                                  GtkTextChildAnchor *anchor);
```

Obtains the location of `anchor` within `buffer` .

Parameters

buffer	a GtkTextBuffer	
iter	an iterator to be initialized.	[out]
anchor	a child anchor that appears in `buffer`	

---

gtk_text_buffer_get_start_iter ()

```
void
gtk_text_buffer_get_start_iter (GtkTextBuffer *buffer,
                                GtkTextIter *iter);
```

Initialized `iter` with the first position in the text buffer. This is the same as using `gtk_text_buffer_get_iter_at_offset()` to get the iter at character offset 0.

Parameters

buffer	a GtkTextBuffer	
iter	iterator to initialize.	[out]

---

gtk_text_buffer_get_end_iter ()

```
void
gtk_text_buffer_get_end_iter (GtkTextBuffer *buffer,
                              GtkTextIter *iter);
```

Initializes `iter` with the “end iterator,” one past the last valid character in the text buffer. If dereferenced with `gtk_text_iter_get_char()`, the end iterator has a character value of 0. The entire buffer lies in the range from the first position in the buffer (call `gtk_text_buffer_get_start_iter()` to get character position 0) to the end iterator.

Parameters

buffer	a GtkTextBuffer	
iter	iterator to initialize.	[out]

---

gtk_text_buffer_get_bounds ()

```
void
gtk_text_buffer_get_bounds (GtkTextBuffer *buffer,
                            GtkTextIter *start,
                            GtkTextIter *end);
```

Retrieves the first and last iterators in the buffer, i.e. the entire buffer lies within the range [`start` ,`end`).

Parameters

buffer	a GtkTextBuffer	
start	iterator to initialize with first position in the buffer.	[out]
end	iterator to initialize with the end iterator.	[out]

---

gtk_text_buffer_get_modified ()

```
gboolean
gtk_text_buffer_get_modified (GtkTextBuffer *buffer);
```

Indicates whether the buffer has been modified since the last call to `gtk_text_buffer_set_modified()` set the modification flag to `FALSE`. Used for example to enable a “save” function in a text editor.

Parameters

buffer	a GtkTextBuffer

Returns

`TRUE` if the buffer has been modified

---

gtk_text_buffer_set_modified ()

```
void
gtk_text_buffer_set_modified (GtkTextBuffer *buffer,
                              gboolean setting);
```

Used to keep track of whether the buffer has been modified since the last time it was saved. Whenever the buffer is saved to disk, call gtk_text_buffer_set_modified (`buffer` , FALSE). When the buffer is modified, it will automatically toggled on the modified bit again. When the modified bit flips, the buffer emits the "modified-changed" signal.

Parameters

buffer	a GtkTextBuffer
setting	modification flag setting

---

gtk_text_buffer_delete_selection ()

```
gboolean
gtk_text_buffer_delete_selection (GtkTextBuffer *buffer,
                                  gboolean interactive,
                                  gboolean default_editable);
```

Deletes the range between the "insert" and "selection_bound" marks, that is, the currently-selected text. If `interactive` is `TRUE`, the editability of the selection will be considered (users can't delete uneditable text).

Parameters

buffer	a GtkTextBuffer
interactive	whether the deletion is caused by user interaction
default_editable	whether the buffer is editable by default

Returns

whether there was a non-empty selection to delete

---

gtk_text_buffer_paste_clipboard ()

```
void
gtk_text_buffer_paste_clipboard (GtkTextBuffer *buffer,
                                 GtkClipboard *clipboard,
                                 GtkTextIter *override_location,
                                 gboolean default_editable);
```

Pastes the contents of a clipboard. If `override_location` is `NULL`, the pasted text will be inserted at the cursor position, or the buffer selection will be replaced if the selection is non-empty.

Note: pasting is asynchronous, that is, we'll ask for the paste data and return, and at some point later after the

main loop runs, the paste data will be inserted.

Parameters

buffer	a GtkTextBuffer	
clipboard	the GtkClipboard to paste from	
override_location	location to insert pasted text, or `NULL`.	[allow-none]
default_editable	whether the buffer is editable by default	

---

gtk_text_buffer_copy_clipboard ()

```
void
gtk_text_buffer_copy_clipboard (GtkTextBuffer *buffer,
                                GtkClipboard *clipboard);
```

Copies the currently-selected text to a clipboard.

Parameters

buffer	a GtkTextBuffer
clipboard	the GtkClipboard object to copy to

---

gtk_text_buffer_cut_clipboard ()

```
void
gtk_text_buffer_cut_clipboard (GtkTextBuffer *buffer,
                               GtkClipboard *clipboard,
                               gboolean default_editable);
```

Copies the currently-selected text to a clipboard, then deletes said text if it’s editable.

Parameters

buffer	a GtkTextBuffer
clipboard	the GtkClipboard object to cut to
default_editable	default editability of the buffer

---

gtk_text_buffer_get_selection_bounds ()

```
gboolean
gtk_text_buffer_get_selection_bounds (GtkTextBuffer *buffer,
                                      GtkTextIter *start,
                                      GtkTextIter *end);
```

Returns `TRUE` if some text is selected; places the bounds of the selection in `start` and `end` (if the selection has

length 0, then `start` and `end` are filled in with the same value). `start` and `end` will be in ascending order. If `start` and `end` are NULL, then they are not filled in, but the return value still indicates whether text is selected.

Parameters

buffer	a GtkTextBuffer a GtkTextBuffer	
start	iterator to initialize with selection start.	[out]
end	iterator to initialize with selection end.	[out]

Returns

whether the selection has nonzero length

---

gtk_text_buffer_begin_user_action ()

```
void
gtk_text_buffer_begin_user_action (GtkTextBuffer *buffer);
```

Called to indicate that the buffer operations between here and a call to `gtk_text_buffer_end_user_action()` are part of a single user-visible operation. The operations between `gtk_text_buffer_begin_user_action()` and `gtk_text_buffer_end_user_action()` can then be grouped when creating an undo stack. GtkTextBuffer maintains a count of calls to `gtk_text_buffer_begin_user_action()` that have not been closed with a call to `gtk_text_buffer_end_user_action()`, and emits the “begin-user-action” and “end-user-action” signals only for the outermost pair of calls. This allows you to build user actions from other user actions.

The “interactive” buffer mutation functions, such as `gtk_text_buffer_insert_interactive()`, automatically call begin/end user action around the buffer operations they perform, so there's no need to add extra calls if you user action consists solely of a single call to one of those functions.

Parameters

buffer	a GtkTextBuffer

---

gtk_text_buffer_end_user_action ()

```
void
gtk_text_buffer_end_user_action (GtkTextBuffer *buffer);
```

Should be paired with a call to `gtk_text_buffer_begin_user_action()`. See that function for a full explanation.

Parameters

buffer	a GtkTextBuffer

---

gtk_text_buffer_add_selection_clipboard ()

```
void
gtk_text_buffer_add_selection_clipboard
                                (GtkTextBuffer *buffer,
                                 GtkClipboard *clipboard);
```

Adds `clipboard` to the list of clipboards in which the selection contents of `buffer` are available. In most cases, `clipboard` will be the GtkClipboard of type `GDK_SELECTION_PRIMARY` for a view of `buffer` .

Parameters

buffer	a GtkTextBuffer
clipboard	a GtkClipboard

---

gtk_text_buffer_remove_selection_clipboard ()

```
void
gtk_text_buffer_remove_selection_clipboard
                                (GtkTextBuffer *buffer,
                                 GtkClipboard *clipboard);
```

Removes a GtkClipboard added with `gtk_text_buffer_add_selection_clipboard()`.

Parameters

buffer	a GtkTextBuffer
clipboard	a GtkClipboard added to `buffer` by `gtk_text_buffer_add_selection_clipboard()`

GtkTextBufferDeserializeFunc ()

```
gboolean
(*GtkTextBufferDeserializeFunc) (GtkTextBuffer *register_buffer,
                                 GtkTextBuffer *content_buffer,
                                 GtkTextIter *iter,
                                 const guint8 *data,
                                 gsize length,
                                 gboolean create_tags,
                                 gpointer user_data,
                                 GError **error);
```

A function that is called to deserialize rich text that has been serialized with `gtk_text_buffer_serialize()`, and insert it at `iter` .

Parameters

register_buffer	the GtkTextBuffer the format is

	registered with	
content_buffer	the GtkTextBuffer to deserialize into	
iter	insertion point for the deserialized text	
data	data to deserialize.	[array length=length]
length	length of `data`	
create_tags	`TRUE` if deserializing may create tags	
user_data	user data that was specified when registering the format	
error	return location for a GError	

Returns

`TRUE` on success, `FALSE` otherwise

---

gtk_text_buffer_deserialize ()

```
gboolean
gtk_text_buffer_deserialize (GtkTextBuffer *register_buffer,
                             GtkTextBuffer *content_buffer,
                             GdkAtom format,
                             GtkTextIter *iter,
                             const guint8 *data,
                             gsize length,
                             GError **error);
```

This function deserializes rich text in format `format` and inserts it at `iter` .

`formats` to be used must be registered using `gtk_text_buffer_register_deserialize_format()` or `gtk_text_buffer_register_deserialize_tagset()` beforehand.

Parameters

register_buffer	the GtkTextBuffer `format` is registered with	
content_buffer	the GtkTextBuffer to deserialize into	
format	the rich text format to use for deserializing	
iter	insertion point for the deserialized text	
data	data to deserialize.	[array length=length]
length	length of `data`	
error	return location for a GError	

Returns

`TRUE` on success, `FALSE` otherwise.

Since: 2.10

---

gtk_text_buffer_deserialize_get_can_create_tags ()

```
gboolean
gtk_text_buffer_deserialize_get_can_create_tags
                                (GtkTextBuffer *buffer,
                                 GdkAtom format);
```

This functions returns the value set with `gtk_text_buffer_deserialize_set_can_create_tags()`

Parameters

buffer	a GtkTextBuffer
format	a GdkAtom representing a registered rich text format

Returns

whether deserializing this format may create tags

Since: 2.10

---

gtk_text_buffer_deserialize_set_can_create_tags ()

```
void
gtk_text_buffer_deserialize_set_can_create_tags
                                (GtkTextBuffer *buffer,
                                 GdkAtom format,
                                 gboolean can_create_tags);
```

Use this function to allow a rich text deserialization function to create new tags in the receiving buffer. Note that using this function is almost always a bad idea, because the rich text functions you register should know how to map the rich text format they handler to your text buffers set of tags.

The ability of creating new (arbitrary!) tags in the receiving buffer is meant for special rich text formats like the internal one that is registered using `gtk_text_buffer_register_deserialize_tagset()`, because that format is essentially a dump of the internal structure of the source buffer, including its tag names.

You should allow creation of tags only if you know what you are doing, e.g. if you defined a tagset name for your application suite's text buffers and you know that it's fine to receive new tags from these buffers, because you know that your application can handle the newly created tags.

Parameters

buffer	a GtkTextBuffer
format	a GdkAtom representing a registered rich text format
can_create_tags	whether deserializing this format may create tags

Since: 2.10

---

gtk_text_buffer_get_copy_target_list ()

```
GtkTargetList *
gtk_text_buffer_get_copy_target_list (GtkTextBuffer *buffer);
```

This function returns the list of targets this text buffer can provide for copying and as DND source. The targets in the list are added with `info` values from the GtkTextBufferTargetInfo enum, using `gtk_target_list_add_rich_text_targets()` and `gtk_target_list_add_text_targets()`.

Parameters

buffer	a GtkTextBuffer	

Returns

the GtkTargetList.

[transfer none]

Since: 2.10

---

gtk_text_buffer_get_deserialize_formats ()

```
GdkAtom *
gtk_text_buffer_get_deserialize_formats
                                (GtkTextBuffer *buffer,
                                 gint *n_formats);
```

This function returns the rich text deserialize formats registered with `buffer` using `gtk_text_buffer_register_deserialize_format()` or `gtk_text_buffer_register_deserialize_tagset()`

Parameters

buffer	a GtkTextBuffer	
n_formats	return location for the number of formats.	[out]

Returns

an array of GdkAtoms representing the registered formats.

[array length=n_formats][transfer container]

Since: 2.10

---

gtk_text_buffer_get_paste_target_list ()

```
GtkTargetList *
gtk_text_buffer_get_paste_target_list (GtkTextBuffer *buffer);
```

This function returns the list of targets this text buffer supports for pasting and as DND destination. The targets in the list are added with `info` values from the GtkTextBufferTargetInfo enum, using `gtk_target_list_add_rich_text_targets()` and `gtk_target_list_add_text_targets()`.

Parameters

buffer	a GtkTextBuffer	

Returns

the GtkTargetList.

[transfer none]

Since: 2.10

---

gtk_text_buffer_get_serialize_formats ()

```
GdkAtom *
gtk_text_buffer_get_serialize_formats (GtkTextBuffer *buffer,
                                       gint *n_formats);
```

This function returns the rich text serialize formats registered with `buffer` using `gtk_text_buffer_register_serialize_format()` or `gtk_text_buffer_register_serialize_tagset()`

Parameters

buffer	a GtkTextBuffer	
n_formats	return location for the number of formats.	[out]

Returns

an array of GdkAtoms representing the registered formats.

[array length=n_formats][transfer container]

Since: 2.10

---

gtk_text_buffer_register_deserialize_format ()

```
GdkAtom
gtk_text_buffer_register_deserialize_format
                                (GtkTextBuffer *buffer,
                                 const gchar *mime_type,
                                 GtkTextBufferDeserializeFunc function,
```

```
                                gpointer user_data,
                                GDestroyNotify user_data_destroy);
```

This function registers a rich text deserialization `function` along with its `mime_type` with the passed `buffer` .

Parameters

buffer	a GtkTextBuffer
mime_type	the format’s mime-type
function	the deserialize function to register
user_data	`function` ’s user_data
user_data_destroy	a function to call when `user_data` is no longer needed

Returns

the GdkAtom that corresponds to the newly registered format’s mime-type.

[transfer none]

Since: 2.10

---

gtk_text_buffer_register_deserialize_tagset ()

```
GdkAtom
gtk_text_buffer_register_deserialize_tagset
                               (GtkTextBuffer *buffer,
                                const gchar *tagset_name);
```

This function registers GTK+’s internal rich text serialization format with the passed `buffer` . See `gtk_text_buffer_register_serialize_tagset()` for details.

Parameters

buffer	a GtkTextBuffer	
tagset_name	an optional tagset name, on `NULL`.	[allow-none]

Returns

the GdkAtom that corresponds to the newly registered format’s mime-type.

[transfer none]

Since: 2.10

---

gtk_text_buffer_register_serialize_format ()

```
GdkAtom
gtk_text_buffer_register_serialize_format
                               (GtkTextBuffer *buffer,
```

```
                                const gchar *mime_type,
                                GtkTextBufferSerializeFunc function,
                                gpointer user_data,
                                GDestroyNotify user_data_destroy);
```

This function registers a rich text serialization `function` along with its `mime_type` with the passed `buffer` .

Parameters

buffer	a GtkTextBuffer
mime_type	the format's mime-type
function	the serialize function to register
user_data	`function` 's user_data
user_data_destroy	a function to call when `user_data` is no longer needed

Returns

the GdkAtom that corresponds to the newly registered format's mime-type.

[transfer none]

Since: 2.10

---

gtk_text_buffer_register_serialize_tagset ()

```
GdkAtom
gtk_text_buffer_register_serialize_tagset
                               (GtkTextBuffer *buffer,
                                const gchar *tagset_name);
```

This function registers GTK+'s internal rich text serialization format with the passed `buffer` . The internal format does not comply to any standard rich text format and only works between GtkTextBuffer instances. It is capable of serializing all of a text buffer's tags and embedded pixbufs.

This function is just a wrapper around `gtk_text_buffer_register_serialize_format()`. The mime type used for registering is "application/x-gtk-text-buffer-rich-text", or "application/x-gtk-text-buffer-rich-text;format=`tagset_name` " if a `tagset_name` was passed.

The `tagset_name` can be used to restrict the transfer of rich text to buffers with compatible sets of tags, in order to avoid unknown tags from being pasted. It is probably the common case to pass an identifier != `NULL` here, since the `NULL` tagset requires the receiving buffer to deal with with pasting of arbitrary tags.

Parameters

buffer	a GtkTextBuffer	
tagset_name	an optional tagset name, on `NULL`.	[allow-none]

Returns

the GdkAtom that corresponds to the newly registered format's mime-type.

[transfer none]

Since: 2.10

GtkTextBufferSerializeFunc ()

```
guint8 *
(*GtkTextBufferSerializeFunc) (GtkTextBuffer *register_buffer,
                               GtkTextBuffer *content_buffer,
                               const GtkTextIter *start,
                               const GtkTextIter *end,
                               gsize *length,
                               gpointer user_data);
```

A function that is called to serialize the content of a text buffer. It must return the serialized form of the content.

Parameters

register_buffer	the GtkTextBuffer for which the format is registered
content_buffer	the GtkTextBuffer to serialize
start	start of the block of text to serialize
end	end of the block of text to serialize
length	Return location for the length of the serialized data
user_data	user data that was specified when registering the format

Returns

a newly-allocated array of guint8 which contains the serialized data, or `NULL` if an error occurred.

[nullable]

---

gtk_text_buffer_serialize ()

```
guint8 *
gtk_text_buffer_serialize (GtkTextBuffer *register_buffer,
                           GtkTextBuffer *content_buffer,
                           GdkAtom format,
                           const GtkTextIter *start,
                           const GtkTextIter *end,
                           gsize *length);
```

This function serializes the portion of text between `start` and `end` in the rich text format represented by `format`.

`formats` to be used must be registered using `gtk_text_buffer_register_serialize_format()` or `gtk_text_buffer_register_serialize_tagset()` beforehand.

Parameters

register_buffer	the GtkTextBuffer `format` is registered with	
content_buffer	the GtkTextBuffer to serialize	
format	the rich text format to use for serializing	
start	start of block of text to serialize	
end	end of block of test to serialize	
length	return location for the length of the serialized data.	[out]

Returns

the serialized data, encoded as `format` .

[array length=length][transfer full]

Since: 2.10

---

gtk_text_buffer_unregister_deserialize_format ()

```
void
gtk_text_buffer_unregister_deserialize_format
                                (GtkTextBuffer *buffer,
                                 GdkAtom format);
```

This function unregisters a rich text format that was previously registered using `gtk_text_buffer_register_deserialize_format()` or `gtk_text_buffer_register_deserialize_tagset()`.

Parameters

buffer	a GtkTextBuffer
format	a GdkAtom representing a registered rich text format.

Since: 2.10

---

gtk_text_buffer_unregister_serialize_format ()

```
void
gtk_text_buffer_unregister_serialize_format
                                (GtkTextBuffer *buffer,
                                 GdkAtom format);
```

This function unregisters a rich text format that was previously registered using `gtk_text_buffer_register_serialize_format()` or `gtk_text_buffer_register_serialize_tagset()`

Parameters

buffer	a GtkTextBuffer
format	a GdkAtom representing a registered rich text format.

Since: 2.10

Types and Values

GtkTextBuffer

```
typedef struct _GtkTextBuffer GtkTextBuffer;
```

---

struct GtkTextBufferClass

```
struct GtkTextBufferClass {
  GObjectClass parent_class;

  void (* insert_text)           (GtkTextBuffer      *buffer,
                                  GtkTextIter        *pos,
                                  const gchar        *new_text,
                                  gint                new_text_length);

  void (* insert_pixbuf)         (GtkTextBuffer      *buffer,
                                  GtkTextIter        *iter,
                                  GdkPixbuf          *pixbuf);

  void (* insert_child_anchor)   (GtkTextBuffer      *buffer,
                                  GtkTextIter        *iter,
                                  GtkTextChildAnchor *anchor);

  void (* delete_range)          (GtkTextBuffer      *buffer,
                                  GtkTextIter        *start,
                                  GtkTextIter        *end);

  void (* changed)               (GtkTextBuffer      *buffer);

  void (* modified_changed)      (GtkTextBuffer      *buffer);

  void (* mark_set)              (GtkTextBuffer      *buffer,
                                  const GtkTextIter  *location,
                                  GtkTextMark        *mark);

  void (* mark_deleted)          (GtkTextBuffer      *buffer,
                                  GtkTextMark        *mark);

  void (* apply_tag)             (GtkTextBuffer      *buffer,
                                  GtkTextTag         *tag,
                                  const GtkTextIter  *start,
                                  const GtkTextIter  *end);

  void (* remove_tag)            (GtkTextBuffer      *buffer,
                                  GtkTextTag         *tag,
                                  const GtkTextIter  *start,
                                  const GtkTextIter  *end);
```

```
  void (* begin_user_action)      (GtkTextBuffer      *buffer);

  void (* end_user_action)        (GtkTextBuffer      *buffer);

  void (* paste_done)             (GtkTextBuffer      *buffer,
                                   GtkClipboard       *clipboard);
};
```

Members

`insert_text` ()	The class handler for the “insert-text” signal.
`insert_pixbuf` ()	The class handler for the “insert-pixbuf” signal.
`insert_child_anchor` ()	The class handler for the “insert-child-anchor” signal.
`delete_range` ()	The class handler for the “delete-range” signal.
`changed` ()	The class handler for the “changed” signal.
`modified_changed` ()	The class handler for the “modified-changed” signal.
`mark_set` ()	The class handler for the “mark-set” signal.
`mark_deleted` ()	The class handler for the “mark-deleted” signal.
`apply_tag` ()	The class handler for the “apply-tag” signal.
`remove_tag` ()	The class handler for the “remove-tag” signal.
`begin_user_action` ()	The class handler for the “begin-user-action” signal.
`end_user_action` ()	The class handler for the “end-user-action” signal.
`paste_done` ()	The class handler for the “paste-done” signal.

---

enum GtkTextBufferTargetInfo

These values are used as “info” for the targets contained in the lists returned by `gtk_text_buffer_get_copy_target_list()` and `gtk_text_buffer_get_paste_target_list()`.

The values counts down from `-1` to avoid clashes with application added drag destinations which usually start at 0.

Members

GTK_TEXT_BUFFER_TARGET_INFO_BUFFER_CONTENTS	Buffer contents

GTK_TEXT_BUFFER_TARGET_INFO_RICH_TEXT	Rich text
GTK_TEXT_BUFFER_TARGET_INFO_TEXT	Text

Property Details

The `"copy-target-list"` property

`"copy-target-list"` `GtkTargetList *`

The list of targets this buffer supports for clipboard copying and as DND source.

Flags: Read

Since: 2.10

The `"cursor-position"` property

`"cursor-position"` `gint`

The position of the insert mark (as offset from the beginning of the buffer). It is useful for getting notified when the cursor moves.

Flags: Read

Allowed values: >= 0

Default value: 0

Since: 2.10

The `"has-selection"` property

`"has-selection"` `gboolean`

Whether the buffer has some text currently selected.

Flags: Read

Default value: FALSE

Since: 2.10

The `"paste-target-list"` property

`"paste-target-list"` `GtkTargetList *`

The list of targets this buffer supports for clipboard pasting and as DND destination.

Flags: Read

Since: 2.10

The “tag-table” property

```
  “tag-table”                GtkTextTagTable *
```

Text Tag Table.

Flags: Read / Write / Construct Only

The “text” property

```
  “text”                     gchar *
```

The text content of the buffer. Without child widgets and images, see `gtk_text_buffer_get_text()` for more information.

Flags: Read / Write

Default value: ""

Since: 2.8

Signal Details

The “apply-tag” signal

```
void
user_function (GtkTextBuffer *textbuffer,
               GtkTextTag    *tag,
               GtkTextIter   *start,
               GtkTextIter   *end,
               gpointer       user_data)
```

The ::apply-tag signal is emitted to apply a tag to a range of text in a GtkTextBuffer. Applying actually occurs in the default handler.

Note that if your handler runs before the default handler it must not invalidate the `start` and `end` iters (or has to revalidate them).

See also: `gtk_text_buffer_apply_tag()`, `gtk_text_buffer_insert_with_tags()`, `gtk_text_buffer_insert_range()`.

Parameters

textbuffer	the object which received the signal
tag	the applied tag
start	the start of the range the tag is applied to
end	the end of the range the tag is applied to
user_data	user data set when the signal

	handler was connected.

Flags: Run Last

The “begin-user-action” signal

```
void
user_function (GtkTextBuffer *textbuffer,
               gpointer       user_data)
```

The ::begin-user-action signal is emitted at the beginning of a single user-visible operation on a GtkTextBuffer.

See also: `gtk_text_buffer_begin_user_action()`, `gtk_text_buffer_insert_interactive()`, `gtk_text_buffer_insert_range_interactive()`, `gtk_text_buffer_delete_interactive()`, `gtk_text_buffer_backspace()`, `gtk_text_buffer_delete_selection()`.

Parameters

textbuffer	the object which received the signal
user_data	user data set when the signal handler was connected.

Flags: Run Last

The “changed” signal

```
void
user_function (GtkTextBuffer *textbuffer,
               gpointer       user_data)
```

The ::changed signal is emitted when the content of a GtkTextBuffer has changed.

Parameters

textbuffer	the object which received the signal
user_data	user data set when the signal handler was connected.

Flags: Run Last

The “delete-range” signal

```
void
user_function (GtkTextBuffer *textbuffer,
               GtkTextIter   *start,
               GtkTextIter   *end,
               gpointer       user_data)
```

The ::delete-range signal is emitted to delete a range from a GtkTextBuffer.

Note that if your handler runs before the default handler it must not invalidate the `start` and `end` iters (or has to revalidate them). The default signal handler revalidates the `start` and `end` iters to both point to the location

where text was deleted. Handlers which run after the default handler (see `g_signal_connect_after()`) do not have access to the deleted text.

See also: `gtk_text_buffer_delete()`.

Parameters

textbuffer	the object which received the signal
start	the start of the range to be deleted
end	the end of the range to be deleted
user_data	user data set when the signal handler was connected.

Flags: Run Last

The `"end-user-action"` signal

```
void
user_function (GtkTextBuffer *textbuffer,
               gpointer       user_data)
```

The ::end-user-action signal is emitted at the end of a single user-visible operation on the GtkTextBuffer.

See also: `gtk_text_buffer_end_user_action()`, `gtk_text_buffer_insert_interactive()`, `gtk_text_buffer_insert_range_interactive()`, `gtk_text_buffer_delete_interactive()`, `gtk_text_buffer_backspace()`, `gtk_text_buffer_delete_selection()`, `gtk_text_buffer_backspace()`.

Parameters

textbuffer	the object which received the signal
user_data	user data set when the signal handler was connected.

Flags: Run Last

The `"insert-child-anchor"` signal

```
void
user_function (GtkTextBuffer      *textbuffer,
               GtkTextIter        *location,
               GtkTextChildAnchor *anchor,
               gpointer            user_data)
```

The ::insert-child-anchor signal is emitted to insert a GtkTextChildAnchor in a GtkTextBuffer. Insertion actually occurs in the default handler.

Note that if your handler runs before the default handler it must not invalidate the `location` iter (or has to revalidate it). The default signal handler revalidates it to be placed after the inserted `anchor` .

See also: `gtk_text_buffer_insert_child_anchor()`.

Parameters

textbuffer	the object which received the signal
location	position to insert `anchor` in `textbuffer`
anchor	the GtkTextChildAnchor to be inserted
user_data	user data set when the signal handler was connected.

Flags: Run Last

The `"insert-pixbuf"` signal

```
void
user_function (GtkTextBuffer *textbuffer,
               GtkTextIter   *location,
               GdkPixbuf     *pixbuf,
               gpointer       user_data)
```

The ::insert-pixbuf signal is emitted to insert a GdkPixbuf in a GtkTextBuffer. Insertion actually occurs in the default handler.

Note that if your handler runs before the default handler it must not invalidate the `location` iter (or has to revalidate it). The default signal handler revalidates it to be placed after the inserted `pixbuf` .

See also: `gtk_text_buffer_insert_pixbuf()`.

Parameters

textbuffer	the object which received the signal
location	position to insert `pixbuf` in `textbuffer`
pixbuf	the GdkPixbuf to be inserted
user_data	user data set when the signal handler was connected.

Flags: Run Last

The `"insert-text"` signal

```
void
user_function (GtkTextBuffer *textbuffer,
               GtkTextIter   *location,
               gchar         *text,
               gint           len,
               gpointer       user_data)
```

The ::insert-text signal is emitted to insert text in a GtkTextBuffer. Insertion actually occurs in the default handler.

Note that if your handler runs before the default handler it must not invalidate the `location` iter (or has to revalidate it). The default signal handler revalidates it to point to the end of the inserted text.

See also: `gtk_text_buffer_insert()`, `gtk_text_buffer_insert_range()`.

Parameters

textbuffer	the object which received the signal
location	position to insert `text` in `textbuffer`
text	the UTF-8 text to be inserted
len	length of the inserted text in bytes
user_data	user data set when the signal handler was connected.

Flags: Run Last

The `"mark-deleted"` signal

```
void
user_function (GtkTextBuffer *textbuffer,
               GtkTextMark   *mark,
               gpointer       user_data)
```

The ::mark-deleted signal is emitted as notification after a GtkTextMark is deleted.

See also: `gtk_text_buffer_delete_mark()`.

Parameters

textbuffer	the object which received the signal
mark	The mark that was deleted
user_data	user data set when the signal handler was connected.

Flags: Run Last

The `"mark-set"` signal

```
void
user_function (GtkTextBuffer *textbuffer,
               GtkTextIter   *location,
               GtkTextMark   *mark,
               gpointer       user_data)
```

The ::mark-set signal is emitted as notification after a GtkTextMark is set.

See also: `gtk_text_buffer_create_mark()`, `gtk_text_buffer_move_mark()`.

Parameters

textbuffer	the object which received the signal
location	The location of `mark` in `textbuffer`
mark	The mark that is set
user_data	user data set when the signal

handler was connected.

Flags: Run Last

The “modified-changed” signal

```
void
user_function (GtkTextBuffer *textbuffer,
               gpointer       user_data)
```

The ::modified-changed signal is emitted when the modified bit of a GtkTextBuffer flips.

See also: gtk_text_buffer_set_modified().

Parameters

textbuffer	the object which received the signal
user_data	user data set when the signal handler was connected.

Flags: Run Last

The “paste-done” signal

```
void
user_function (GtkTextBuffer *textbuffer,
               GtkClipboard  *clipboard,
               gpointer       user_data)
```

The paste-done signal is emitted after paste operation has been completed. This is useful to properly scroll the view to the end of the pasted text. See gtk_text_buffer_paste_clipboard() for more details.

Parameters

textbuffer	the object which received the signal
clipboard	the GtkClipboard pasted from
user_data	user data set when the signal handler was connected.

Flags: Run Last

Since: 2.16

The “remove-tag” signal

```
void
user_function (GtkTextBuffer *textbuffer,
               GtkTextTag    *tag,
               GtkTextIter   *start,
               GtkTextIter   *end,
               gpointer       user_data)
```

The ::remove-tag signal is emitted to remove all occurrences of `tag` from a range of text in a GtkTextBuffer.

Removal actually occurs in the default handler.

Note that if your handler runs before the default handler it must not invalidate the `start` and `end` iters (or has to revalidate them).

See also: `gtk_text_buffer_remove_tag()`.

Parameters

textbuffer	the object which received the signal
tag	the tag to be removed
start	the start of the range the tag is removed from
end	the end of the range the tag is removed from
user_data	user data set when the signal handler was connected.

Flags: Run Last

See Also

GtkTextView, GtkTextIter, GtkTextMark

GtkTextTag

GtkTextTag — A tag that can be applied to text in a GtkTextBuffer

Functions

GtkTextTag *	gtk_text_tag_new ()
gint	gtk_text_tag_get_priority ()
void	gtk_text_tag_set_priority ()
gboolean	gtk_text_tag_event ()
void	gtk_text_tag_changed ()
GtkTextAttributes *	gtk_text_attributes_new ()
GtkTextAttributes *	gtk_text_attributes_copy ()
void	gtk_text_attributes_copy_values ()
void	gtk_text_attributes_unref ()
GtkTextAttributes *	gtk_text_attributes_ref ()

Properties

gboolean	accumulative-margin	Read / Write
gchar *	background	Write
gboolean	background-full-height	Read / Write
gboolean	background-full-height-set	Read / Write

GdkColor *	background-gdk	Read / Write
GdkRGBA *	background-rgba	Read / Write
gboolean	background-set	Read / Write
GtkTextDirection	direction	Read / Write
gboolean	editable	Read / Write
gboolean	editable-set	Read / Write
gboolean	fallback	Read / Write
gboolean	fallback-set	Read / Write
gchar *	family	Read / Write
gboolean	family-set	Read / Write
gchar *	font	Read / Write
PangoFontDescription *	font-desc	Read / Write
gchar *	font-features	Read / Write
gboolean	font-features-set	Read / Write
gchar *	foreground	Write
GdkColor *	foreground-gdk	Read / Write
GdkRGBA *	foreground-rgba	Read / Write
gboolean	foreground-set	Read / Write
gint	indent	Read / Write
gboolean	indent-set	Read / Write
gboolean	invisible	Read / Write
gboolean	invisible-set	Read / Write
GtkJustification	justification	Read / Write
gboolean	justification-set	Read / Write
gchar *	language	Read / Write
gboolean	language-set	Read / Write
gint	left-margin	Read / Write
gboolean	left-margin-set	Read / Write
gint	letter-spacing	Read / Write
gboolean	letter-spacing-set	Read / Write
gchar *	name	Read / Write / Construct Only
gchar *	paragraph-background	Write
GdkColor *	paragraph-background-gdk	Read / Write
GdkRGBA *	paragraph-background-rgba	Read / Write
gboolean	paragraph-background-set	Read / Write
gint	pixels-above-lines	Read / Write
gboolean	pixels-above-lines-set	Read / Write
gint	pixels-below-lines	Read / Write
gboolean	pixels-below-lines-set	Read / Write
gint	pixels-inside-wrap	Read / Write
gboolean	pixels-inside-wrap-set	Read / Write
gint	right-margin	Read / Write
gboolean	right-margin-set	Read / Write
gint	rise	Read / Write
gboolean	rise-set	Read / Write
gdouble	scale	Read / Write
gboolean	scale-set	Read / Write
gint	size	Read / Write
gdouble	size-points	Read / Write
gboolean	size-set	Read / Write
PangoStretch	stretch	Read / Write
gboolean	stretch-set	Read / Write

gboolean	strikethrough	Read / Write
GdkRGBA *	strikethrough-rgba	Read / Write
gboolean	strikethrough-rgba-set	Read / Write
gboolean	strikethrough-set	Read / Write
PangoStyle	style	Read / Write
gboolean	style-set	Read / Write
PangoTabArray *	tabs	Read / Write
gboolean	tabs-set	Read / Write
PangoUnderline	underline	Read / Write
GdkRGBA *	underline-rgba	Read / Write
gboolean	underline-rgba-set	Read / Write
gboolean	underline-set	Read / Write
PangoVariant	variant	Read / Write
gboolean	variant-set	Read / Write
gint	weight	Read / Write
gboolean	weight-set	Read / Write
GtkWrapMode	wrap-mode	Read / Write
gboolean	wrap-mode-set	Read / Write

Signals

gboolean	event	Run Last

Types and Values

struct	GtkTextTag
struct	GtkTextAttributes
struct	GtkTextAppearance

Object Hierarchy

```
GObject
└── GtkTextTag
```

Includes

```
#include <gtk/gtk.h>
```

Description

You may wish to begin by reading the text widget conceptual overview which gives an overview of all the objects and data types related to the text widget and how they work together.

Tags should be in the GtkTextTagTable for a given GtkTextBuffer before using them with that buffer.

gtk_text_buffer_create_tag() is the best way to create tags. See “gtk3-demo” for numerous examples.

For each property of GtkTextTag, there is a “set” property, e.g. “font-set” corresponds to “font”. These “set” properties reflect whether a property has been set or not. They are maintained by GTK+ and you should not set them independently.

Functions

gtk_text_tag_new ()

```
GtkTextTag *
gtk_text_tag_new (const gchar *name);
```

Creates a GtkTextTag. Configure the tag using object arguments, i.e. using `g_object_set()`.

Parameters

name	tag name, or `NULL`.	[allow-none]

Returns

a new GtkTextTag

---

gtk_text_tag_get_priority ()

```
gint
gtk_text_tag_get_priority (GtkTextTag *tag);
```

Get the tag priority.

Parameters

tag	a GtkTextTag

Returns

The tag's priority.

---

gtk_text_tag_set_priority ()

```
void
gtk_text_tag_set_priority (GtkTextTag *tag,
                           gint priority);
```

Sets the priority of a GtkTextTag. Valid priorities start at 0 and go to one less than `gtk_text_tag_table_get_size()`. Each tag in a table has a unique priority; setting the priority of one tag shifts the priorities of all the other tags in the table to maintain a unique priority for each tag. Higher priority tags "win" if two tags both set the same text attribute. When adding a tag to a tag table, it will be assigned the highest priority in the table by default; so normally the precedence of a set of tags is the order in which they were added to the table, or created with `gtk_text_buffer_create_tag()`, which adds the tag to the buffer's table automatically.

Parameters

tag	a GtkTextTag
priority	the new priority

---

gtk_text_tag_event ()

```
gboolean
gtk_text_tag_event (GtkTextTag *tag,
                    GObject *event_object,
                    GdkEvent *event,
                    const GtkTextIter *iter);
```

Emits the “event” signal on the GtkTextTag.

Parameters

tag	a GtkTextTag
event_object	object that received the event, such as a widget
event	the event
iter	location where the event was received

Returns

result of signal emission (whether the event was handled)

---

gtk_text_tag_changed ()

```
void
gtk_text_tag_changed (GtkTextTag *tag,
                      gboolean size_changed);
```

Emits the “tag-changed” signal on the GtkTextTagTable where the tag is included.

The signal is already emitted when setting a GtkTextTag property. This function is useful for a GtkTextTag subclass.

Parameters

tag	a GtkTextTag.
size_changed	whether the change affects the GtkTextView layout.

Since: 3.20

---

gtk_text_attributes_new ()

```
GtkTextAttributes *
gtk_text_attributes_new (void);
```

Creates a GtkTextAttributes, which describes a set of properties on some text.

Returns

a new GtkTextAttributes, free with `gtk_text_attributes_unref()`.

---

gtk_text_attributes_copy ()

```
GtkTextAttributes *
gtk_text_attributes_copy (GtkTextAttributes *src);
```

Copies `src` and returns a new GtkTextAttributes.

Parameters

src	a GtkTextAttributes to be copied

Returns

a copy of `src` , free with `gtk_text_attributes_unref()`

---

gtk_text_attributes_copy_values ()

```
void
gtk_text_attributes_copy_values (GtkTextAttributes *src,
                                 GtkTextAttributes *dest);
```

Copies the values from `src` to `dest` so that `dest` has the same values as `src` . Frees existing values in `dest` .

Parameters

src	a GtkTextAttributes
dest	another GtkTextAttributes

---

gtk_text_attributes_unref ()

```
void
gtk_text_attributes_unref (GtkTextAttributes *values);
```

Decrements the reference count on `values` , freeing the structure if the reference count reaches 0.

Parameters

values	a GtkTextAttributes

---

gtk_text_attributes_ref ()

```
GtkTextAttributes *
gtk_text_attributes_ref (GtkTextAttributes *values);
```

Increments the reference count on `values` .

Parameters

values	a GtkTextAttributes

Returns

the GtkTextAttributes that were passed in

Types and Values

struct GtkTextTag

```
struct GtkTextTag;
```

---

struct GtkTextAttributes

```
struct GtkTextAttributes {
  GtkTextAppearance appearance;

  GtkJustification justification;
  GtkTextDirection direction;

  PangoFontDescription *font;

  gdouble font_scale;

  gint left_margin;
  gint right_margin;
  gint indent;

  gint pixels_above_lines;
  gint pixels_below_lines;
  gint pixels_inside_wrap;

  PangoTabArray *tabs;

  GtkWrapMode wrap_mode;

  PangoLanguage *language;
```

```
  guint invisible : 1;
  guint bg_full_height : 1;
  guint editable : 1;
  guint no_fallback: 1;

  gint letter_spacing;

#ifdef __GI_SCANNER__
  /* The scanner should only see the transparent union, so that its
   * content does not vary across architectures.
   */
  union {
    gchar *font_features;
};
```

Using GtkTextAttributes directly should rarely be necessary. It's primarily useful with `gtk_text_iter_get_attributes()`. As with most GTK+ structs, the fields in this struct should only be read, never modified directly.

Members

GtkTextAppearance `appearance`;	GtkTextAppearance for text.
GtkJustification `justification`;	GtkJustification for text.
GtkTextDirection `direction`;	GtkTextDirection for text.
PangoFontDescription *`font`;	PangoFontDescription for text.
gdouble `font_scale`;	Font scale factor.
gint `left_margin`;	Width of the left margin in pixels.
gint `right_margin`;	Width of the right margin in pixels.
gint `indent`;	Amount to indent the paragraph, in pixels.
gint `pixels_above_lines`;	Pixels of blank space above paragraphs.
gint `pixels_below_lines`;	Pixels of blank space below paragraphs.
gint `pixels_inside_wrap`;	Pixels of blank space between wrapped lines in a paragraph.
PangoTabArray *`tabs`;	Custom PangoTabArray for this text.
GtkWrapMode `wrap_mode`;	GtkWrapMode for text.
PangoLanguage *`language`;	PangoLanguage for text.
guint `invisible` : 1;	Hide the text.
guint `bg_full_height` : 1;	Background is fit to full line height rather than baseline +/- ascent/descent (font height).
guint `editable` : 1;	Can edit this text.
guint `no_fallback` : 1;	Whether to disable font fallback.
gint `letter_spacing`;	Extra space to insert between graphemes, in Pango units
gchar *`font_features`;	

struct GtkTextAppearance

```
struct GtkTextAppearance {
  GdkColor bg_color; /* pixel is taken for underline color */
  GdkColor fg_color; /* pixel is taken for strikethrough color */

  /* super/subscript rise, can be negative */
  gint rise;

  guint underline : 4;          /* PangoUnderline */
  guint strikethrough : 1;

  /* Whether to use background-related values; this is irrelevant for
   * the values struct when in a tag, but is used for the composite
   * values struct; it's true if any of the tags being composited
   * had background stuff set.
   */
  guint draw_bg : 1;

  /* These are only used when we are actually laying out and rendering
   * a paragraph; not when a GtkTextAppearance is part of a
   * GtkTextAttributes.
   */
  guint inside_selection : 1;
  guint is_text : 1;

  /* For the sad story of this bit of code, see
   * https://bugzilla.gnome.org/show_bug.cgi?id=711158
   */
#ifdef __GI_SCANNER__
  /* The scanner should only see the transparent union, so that its
   * content does not vary across architectures.
   */
  union {
    GdkRGBA *rgba[2];
};
```

Members

GdkColor `bg_color`;	Background GdkColor.
GdkColor `fg_color`;	Foreground GdkColor.
gint `rise`;	Super/subscript rise, can be negative.
guint `underline` : 4;	PangoUnderline
guint `strikethrough` : 1;	Strikethrough style
guint `draw_bg` : 1;	Whether to use background-related values; this is irrelevant for the values struct when in a tag, but is used for the composite values struct; it’s true if any of the tags being composited had background stuff set.
guint `inside_selection` : 1;	This are only used when we are actually laying out and rendering a paragraph; not when a GtkTextAppearance is part of a GtkTextAttributes.
guint `is_text` : 1;	This are only used when we are

	actually laying out and rendering a paragraph; not when a GtkTextAppearance is part of a GtkTextAttributes.
GdkRGBA *rgba[2];	GdkRGBA

Property Details

The “accumulative-margin” property

“accumulative-margin” gboolean

Whether the margins accumulate or override each other.

When set to TRUE the margins of this tag are added to the margins of any other non-accumulative margins present. When set to FALSE the margins override one another (the default).

Flags: Read / Write

Default value: FALSE

Since: 2.12

The “background” property

“background” gchar *

Background color as a string.

Flags: Write

Default value: NULL

The “background-full-height” property

“background-full-height” gboolean

Whether the background color fills the entire line height or only the height of the tagged characters.

Flags: Read / Write

Default value: FALSE

The “background-full-height-set” property

“background-full-height-set” gboolean

Whether this tag affects background height.

Flags: Read / Write

Default value: FALSE

The “background-gdk” property

“background-gdk” GdkColor *

Background color as a GdkColor.

GtkTextTag:background-gdk has been deprecated since version 3.4 and should not be used in newly-written code.

Use “background-rgba” instead.

Flags: Read / Write

The “background-rgba” property

“background-rgba” GdkRGBA *

Background color as a GdkRGBA.

Flags: Read / Write

Since: 3.2

The “background-set” property

“background-set” gboolean

Whether this tag affects the background color.

Flags: Read / Write

Default value: FALSE

The “direction” property

“direction” GtkTextDirection

Text direction, e.g. right-to-left or left-to-right.

Flags: Read / Write

Default value: GTK_TEXT_DIR_NONE

The “editable” property

“editable” gboolean

Whether the text can be modified by the user.

Flags: Read / Write

Default value: TRUE

The “editable-set” property

“editable-set” gboolean

Whether this tag affects text editability.

Flags: Read / Write

Default value: FALSE

The “fallback” property

“fallback” gboolean

Whether font fallback is enabled.

When set to TRUE, other fonts will be substituted where the current font is missing glyphs.

Flags: Read / Write

Default value: TRUE

Since: 3.16

The “fallback-set” property

“fallback-set” gboolean

Whether this tag affects font fallback.

Flags: Read / Write

Default value: FALSE

The “family” property

“family” gchar *

Name of the font family, e.g. Sans, Helvetica, Times, Monospace.

Flags: Read / Write

Default value: NULL

The “family-set” property

“family-set” gboolean

Whether this tag affects the font family.

Flags: Read / Write

Default value: FALSE

The “font” property

“font” gchar *

Font description as string, e.g. \"Sans Italic 12\".

Note that the initial value of this property depends on the internals of PangoFontDescription.

Flags: Read / Write

Default value: NULL

The “font-desc” property

“font-desc” PangoFontDescription *

Font description as a PangoFontDescription struct.

Flags: Read / Write

The “font-features” property

“font-features” gchar *

OpenType font features, as a string.

Flags: Read / Write

Default value: NULL

Since: 3.18

The “font-features-set” property

“font-features-set” gboolean

Whether this tag affects font features.

Flags: Read / Write

Default value: FALSE

The “foreground” property

“foreground” gchar *

Foreground color as a string.

Flags: Write

Default value: NULL

The “foreground-gdk” property

“foreground-gdk” GdkColor *

Foreground color as a GdkColor.

GtkTextTag:foreground-gdk has been deprecated since version 3.4 and should not be used in newly-written code.

Use “foreground-rgba” instead.

Flags: Read / Write

The “foreground-rgba” property

“foreground-rgba” GdkRGBA *

Foreground color as a GdkRGBA.

Flags: Read / Write

Since: 3.2

The “foreground-set” property

“foreground-set” gboolean

Whether this tag affects the foreground color.

Flags: Read / Write

Default value: FALSE

The “indent” property

“indent” gint

Amount to indent the paragraph, in pixels.

Flags: Read / Write

Default value: 0

The “indent-set” property

“indent-set” gboolean

Whether this tag affects indentation.

Flags: Read / Write

Default value: FALSE

The “invisible” property

“invisible” gboolean

Whether this text is hidden.

Note that there may still be problems with the support for invisible text, in particular when navigating programmatically inside a buffer containing invisible segments.

Flags: Read / Write

Default value: FALSE

Since: 2.8

The “invisible-set” property

“invisible-set” gboolean

Whether this tag affects text visibility.

Flags: Read / Write

Default value: FALSE

The “justification” property

“justification” GtkJustification

Left, right, or center justification.

Flags: Read / Write

Default value: GTK_JUSTIFY_LEFT

The “justification-set” property

“justification-set” gboolean

Whether this tag affects paragraph justification.

Flags: Read / Write

Default value: FALSE

The “language” property

“language” gchar *

The language this text is in, as an ISO code. Pango can use this as a hint when rendering the text. If not set, an appropriate default will be used.

Note that the initial value of this property depends on the current locale, see also gtk_get_default_language().

Flags: Read / Write

Default value: NULL

The “language-set” property

“language-set” gboolean

Whether this tag affects the language the text is rendered as.

Flags: Read / Write

Default value: FALSE

The “left-margin” property

“left-margin” gint

Width of the left margin in pixels.

Flags: Read / Write

Allowed values: >= 0

Default value: 0

The “left-margin-set” property

“left-margin-set” gboolean

Whether this tag affects the left margin.

Flags: Read / Write

Default value: FALSE

The “letter-spacing” property

“letter-spacing” gint

Extra spacing between graphemes, in Pango units.

Flags: Read / Write

Allowed values: >= 0

Default value: 0

Since: 3.16

The “letter-spacing-set” property

“letter-spacing-set” gboolean

Whether this tag affects letter spacing.

Flags: Read / Write

Default value: FALSE

The “name” property

“name” gchar *

Name used to refer to the text tag. NULL for anonymous tags.

Flags: Read / Write / Construct Only

Default value: NULL

The “paragraph-background” property

“paragraph-background” gchar *

The paragraph background color as a string.

Flags: Write

Default value: NULL

Since: 2.8

The “paragraph-background-gdk” property

“paragraph-background-gdk” GdkColor *

The paragraph background color as a GdkColor.

GtkTextTag:paragraph-background-gdk has been deprecated since version 3.4 and should not be used in newly-written code.

Use “paragraph-background-rgba” instead.

Flags: Read / Write

Since: 2.8

The “paragraph-background-rgba” property

“paragraph-background-rgba” GdkRGBA *

The paragraph background color as a GdkRGBA.

Flags: Read / Write

Since: 3.2

The “paragraph-background-set” property

“paragraph-background-set” gboolean

Whether this tag affects the paragraph background color.

Flags: Read / Write

Default value: FALSE

The “pixels-above-lines” property

“pixels-above-lines” gint

Pixels of blank space above paragraphs.

Flags: Read / Write

Allowed values: >= 0

Default value: 0

The “pixels-above-lines-set” property

“pixels-above-lines-set” gboolean

Whether this tag affects the number of pixels above lines.

Flags: Read / Write

Default value: FALSE

The “pixels-below-lines” property

“pixels-below-lines” gint

Pixels of blank space below paragraphs.

Flags: Read / Write

Allowed values: >= 0

Default value: 0

The “pixels-below-lines-set” property

“pixels-below-lines-set” gboolean

Whether this tag affects the number of pixels above lines.

Flags: Read / Write

Default value: FALSE

The “pixels-inside-wrap” property

“pixels-inside-wrap” gint

Pixels of blank space between wrapped lines in a paragraph.

Flags: Read / Write

Allowed values: >= 0

Default value: 0

The “pixels-inside-wrap-set” property

“pixels-inside-wrap-set” gboolean

Whether this tag affects the number of pixels between wrapped lines.

Flags: Read / Write

Default value: FALSE

The “right-margin” property

“right-margin” gint

Width of the right margin in pixels.

Flags: Read / Write

Allowed values: >= 0

Default value: 0

The “right-margin-set” property

“right-margin-set” gboolean

Whether this tag affects the right margin.

Flags: Read / Write

Default value: FALSE

The “rise” property

“rise” gint

Offset of text above the baseline (below the baseline if rise is negative) in Pango units.

Flags: Read / Write

Default value: 0

The “rise-set” property

“rise-set” gboolean

Whether this tag affects the rise.

Flags: Read / Write

Default value: FALSE

The “scale” property

“scale” gdouble

Font size as a scale factor relative to the default font size. This properly adapts to theme changes etc. so is recommended. Pango predefines some scales such as PANGO_SCALE_X_LARGE.

Flags: Read / Write

Allowed values: >= 0

Default value: 1

The “scale-set” property

“scale-set” gboolean

Whether this tag scales the font size by a factor.

Flags: Read / Write

Default value: FALSE

The “size” property

“size” gint

Font size in Pango units.

Flags: Read / Write

Allowed values: >= 0

Default value: 0

The “size-points” property

“size-points” gdouble

Font size in points.

Flags: Read / Write

Allowed values: >= 0

Default value: 0

The “size-set” property

“size-set” gboolean

Whether this tag affects the font size.

Flags: Read / Write

Default value: FALSE

The “stretch” property

“stretch” PangoStretch

Font stretch as a PangoStretch, e.g. PANGO_STRETCH_CONDENSED.

Flags: Read / Write

Default value: PANGO_STRETCH_NORMAL

The “stretch-set” property

“stretch-set” gboolean

Whether this tag affects the font stretch.

Flags: Read / Write

Default value: FALSE

The “strikethrough” property

“strikethrough” gboolean

Whether to strike through the text.

Flags: Read / Write

Default value: FALSE

The “strikethrough-rgba” property

“strikethrough-rgba” GdkRGBA *

This property modifies the color of strikeouts. If not set, strikeouts will use the forground color.

Flags: Read / Write

Since: 3.16

The “strikethrough-rgba-set” property

“strikethrough-rgba-set” gboolean

If the “strikethrough-rgba” property has been set.

Flags: Read / Write

Default value: FALSE

Since: 3.16

The “strikethrough-set” property

“strikethrough-set” gboolean

Whether this tag affects strikethrough.

Flags: Read / Write

Default value: FALSE

The “style” property

“style” PangoStyle

Font style as a PangoStyle, e.g. PANGO_STYLE_ITALIC.

Flags: Read / Write

Default value: PANGO_STYLE_NORMAL

The “style-set” property

“style-set” gboolean

Whether this tag affects the font style.

Flags: Read / Write

Default value: FALSE

The “tabs” property

“tabs” PangoTabArray *

Custom tabs for this text.

Flags: Read / Write

The “tabs-set” property

“tabs-set” gboolean

Whether this tag affects tabs.

Flags: Read / Write

Default value: FALSE

The “underline” property

“underline” PangoUnderline

Style of underline for this text.

Flags: Read / Write

Default value: PANGO_UNDERLINE_NONE

The “underline-rgba” property

“underline-rgba” GdkRGBA *

This property modifies the color of underlines. If not set, underlines will use the forground color.

If “underline” is set to PANGO_UNDERLINE_ERROR, an alternate color may be applied instead of the foreground. Setting this property will always override those defaults.

Flags: Read / Write

Since: 3.16

The “underline-rgba-set” property

“underline-rgba-set” gboolean

If the “underline-rgba” property has been set.

Flags: Read / Write

Default value: FALSE

Since: 3.16

The “underline-set” property

“underline-set” gboolean

Whether this tag affects underlining.

Flags: Read / Write

Default value: FALSE

The “variant” property

“variant” PangoVariant

Font variant as a PangoVariant, e.g. PANGO_VARIANT_SMALL_CAPS.

Flags: Read / Write

Default value: PANGO_VARIANT_NORMAL

The “variant-set” property

“variant-set” gboolean

Whether this tag affects the font variant.

Flags: Read / Write

Default value: FALSE

The “weight” property

“weight” gint

Font weight as an integer, see predefined values in PangoWeight; for example, PANGO_WEIGHT_BOLD.

Flags: Read / Write

Allowed values: >= 0

Default value: 400

The “weight-set” property

“weight-set” gboolean

Whether this tag affects the font weight.

Flags: Read / Write

Default value: FALSE

The “wrap-mode” property

“wrap-mode” GtkWrapMode

Whether to wrap lines never, at word boundaries, or at character boundaries.

Flags: Read / Write

Default value: GTK_WRAP_NONE

The “wrap-mode-set” property

“wrap-mode-set” gboolean

Whether this tag affects line wrap mode.

Flags: Read / Write

Default value: FALSE

Signal Details

The “event” signal

```
gboolean
user_function (GtkTextTag  *tag,
               GObject     *object,
               GdkEvent    *event,
               GtkTextIter *iter,
               gpointer     user_data)
```

The ::event signal is emitted when an event occurs on a region of the buffer marked with this tag.

Parameters

tag	the GtkTextTag on which the signal is emitted
object	the object the event was fired from (typically a GtkTextView)
event	the event which triggered the signal

iter	a GtkTextIter pointing at the location the event occurred
user_data	user data set when the signal handler was connected.

Returns

TRUE to stop other handlers from being invoked for the event. FALSE to propagate the event further.

Flags: Run Last

GtkTextTagTable

GtkTextTagTable — Collection of tags that can be used together

Functions

void	(*GtkTextTagTableForeach) ()
GtkTextTagTable *	gtk_text_tag_table_new ()
gboolean	gtk_text_tag_table_add ()
void	gtk_text_tag_table_remove ()
GtkTextTag *	gtk_text_tag_table_lookup ()
void	gtk_text_tag_table_foreach ()
gint	gtk_text_tag_table_get_size ()

Signals

void	tag-added	Run Last
void	tag-changed	Run Last
void	tag-removed	Run Last

Types and Values

GtkTextTagTable

Object Hierarchy

```
GObject
└── GtkTextTagTable
```

Implemented Interfaces

GtkTextTagTable implements GtkBuildable.

Includes

```
#include <gtk/gtk.h>
```

Description

You may wish to begin by reading the text widget conceptual overview which gives an overview of all the objects and data types related to the text widget and how they work together.

GtkTextTagTables as GtkBuildable

The GtkTextTagTable implementation of the GtkBuildable interface supports adding tags by specifying “tag” as the “type” attribute of a <child> element.

An example of a UI definition fragment specifying tags:

```
<object class="GtkTextTagTable">
 <child type="tag">
   <object class="GtkTextTag"/>
 </child>
</object>
```

Functions

GtkTextTagTableForeach ()

```
void
(*GtkTextTagTableForeach) (GtkTextTag *tag,
                           gpointer data);
```

Parameters

tag	the GtkTextTag	
data	data passed to `gtk_text_tag_table_foreach()`.	[closure]

---

gtk_text_tag_table_new ()

```
GtkTextTagTable *
gtk_text_tag_table_new (void);
```

Creates a new GtkTextTagTable. The table contains no tags by default.

Returns

a new GtkTextTagTable

---

gtk_text_tag_table_add ()

```
gboolean
gtk_text_tag_table_add (GtkTextTagTable *table,
                        GtkTextTag *tag);
```

Add a tag to the table. The tag is assigned the highest priority in the table.

`tag` must not be in a tag table already, and may not have the same name as an already-added tag.

Parameters

table	a GtkTextTagTable
tag	a GtkTextTag

Returns

`TRUE` on success.

---

gtk_text_tag_table_remove ()

```
void
gtk_text_tag_table_remove (GtkTextTagTable *table,
                           GtkTextTag *tag);
```

Remove a tag from the table. If a GtkTextBuffer has `table` as its tag table, the tag is removed from the buffer. The table's reference to the tag is removed, so the tag will end up destroyed if you don't have a reference to it.

Parameters

table	a GtkTextTagTable
tag	a GtkTextTag

---

gtk_text_tag_table_lookup ()

```
GtkTextTag *
gtk_text_tag_table_lookup (GtkTextTagTable *table,
                           const gchar *name);
```

Look up a named tag.

Parameters

table	a GtkTextTagTable
name	name of a tag

Returns

The tag, or `NULL` if none by that name is in the table.

[nullable][transfer none]

---

gtk_text_tag_table_foreach ()

```
void
gtk_text_tag_table_foreach (GtkTextTagTable *table,
                            GtkTextTagTableForeach func,
                            gpointer data);
```

Calls `func` on each tag in `table` , with user data `data` . Note that the table may not be modified while iterating over it (you can’t add/remove tags).

Parameters

table	a GtkTextTagTable	
func	a function to call on each tag.	[scope call]
data	user data	

---

gtk_text_tag_table_get_size ()

```
gint
gtk_text_tag_table_get_size (GtkTextTagTable *table);
```

Returns the size of the table (number of tags)

Parameters

table	a GtkTextTagTable

Returns

number of tags in `table`

Types and Values

GtkTextTagTable

```
typedef struct _GtkTextTagTable GtkTextTagTable;
```

Signal Details

The “tag-added” signal

```
void
```

```
user_function (GtkTextTagTable *texttagtable,
               GtkTextTag      *tag,
               gpointer         user_data)
```

Parameters

texttagtable	the object which received the signal.
tag	the added tag.
user_data	user data set when the signal handler was connected.

Flags: Run Last

The `"tag-changed"` signal

```
void
user_function (GtkTextTagTable *texttagtable,
               GtkTextTag      *tag,
               gboolean         size_changed,
               gpointer         user_data)
```

Parameters

texttagtable	the object which received the signal.
tag	the changed tag.
size_changed	whether the change affects the GtkTextView layout.
user_data	user data set when the signal handler was connected.

Flags: Run Last

The `"tag-removed"` signal

```
void
user_function (GtkTextTagTable *texttagtable,
               GtkTextTag      *tag,
               gpointer         user_data)
```

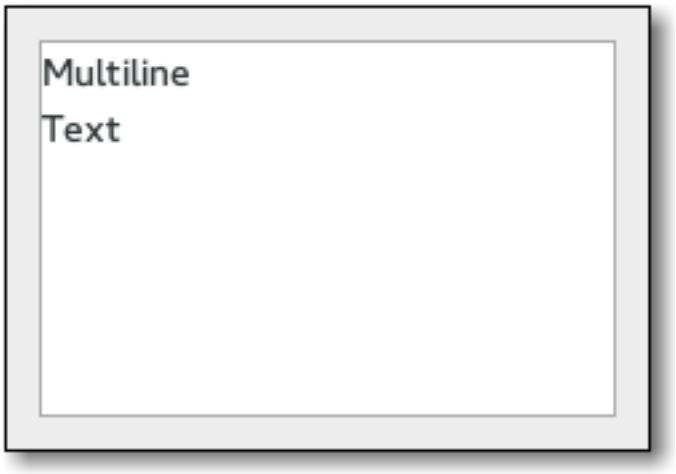
Parameters

texttagtable	the object which received the signal.
tag	the removed tag.
user_data	user data set when the signal handler was connected.

Flags: Run Last

GtkTextView

GtkTextView — Widget that displays a GtkTextBuffer



Functions

GtkWidget *	gtk_text_view_new ()
GtkWidget *	gtk_text_view_new_with_buffer ()
void	gtk_text_view_set_buffer ()
GtkTextBuffer *	gtk_text_view_get_buffer ()
GtkAdjustment *	gtk_text_view_get_hadjustment ()
GtkAdjustment *	gtk_text_view_get_vadjustment ()
void	gtk_text_view_scroll_to_mark ()
gboolean	gtk_text_view_scroll_to_iter ()
void	gtk_text_view_scroll_mark_onscreen ()
gboolean	gtk_text_view_move_mark_onscreen ()
gboolean	gtk_text_view_place_cursor_onscreen ()
void	gtk_text_view_get_visible_rect ()
void	gtk_text_view_get_iter_location ()
void	gtk_text_view_get_cursor_locations ()
void	gtk_text_view_get_line_at_y ()
void	gtk_text_view_get_line_yrange ()
gboolean	gtk_text_view_get_iter_at_location ()
gboolean	gtk_text_view_get_iter_at_position ()
void	gtk_text_view_buffer_to_window_coords ()
void	gtk_text_view_window_to_buffer_coords ()
GdkWindow *	gtk_text_view_get_window ()
GtkTextWindowType	gtk_text_view_get_window_type ()
void	gtk_text_view_set_border_window_size ()
gint	gtk_text_view_get_border_window_size ()
gboolean	gtk_text_view_forward_display_line ()
gboolean	gtk_text_view_backward_display_line ()
gboolean	gtk_text_view_forward_display_line_end ()
gboolean	gtk_text_view_backward_display_line_start ()
gboolean	gtk_text_view_starts_display_line ()
gboolean	gtk_text_view_move_visually ()
void	gtk_text_view_add_child_at_anchor ()
GtkTextChildAnchor *	gtk_text_child_anchor_new ()
GList *	gtk_text_child_anchor_get_widgets ()
gboolean	gtk_text_child_anchor_get_deleted ()
void	gtk_text_view_add_child_in_window ()

void	gtk_text_view_move_child ()
void	gtk_text_view_set_wrap_mode ()
GtkWrapMode	gtk_text_view_get_wrap_mode ()
void	gtk_text_view_set_editable ()
gboolean	gtk_text_view_get_editable ()
void	gtk_text_view_set_cursor_visible ()
gboolean	gtk_text_view_get_cursor_visible ()
void	gtk_text_view_reset_cursor_blink ()
void	gtk_text_view_set_overwrite ()
gboolean	gtk_text_view_get_overwrite ()
void	gtk_text_view_set_pixels_above_lines ()
gint	gtk_text_view_get_pixels_above_lines ()
void	gtk_text_view_set_pixels_below_lines ()
gint	gtk_text_view_get_pixels_below_lines ()
void	gtk_text_view_set_pixels_inside_wrap ()
gint	gtk_text_view_get_pixels_inside_wrap ()
void	gtk_text_view_set_justification ()
GtkJustification	gtk_text_view_get_justification ()
void	gtk_text_view_set_left_margin ()
gint	gtk_text_view_get_left_margin ()
void	gtk_text_view_set_right_margin ()
gint	gtk_text_view_get_right_margin ()
void	gtk_text_view_set_top_margin ()
gint	gtk_text_view_get_top_margin ()
void	gtk_text_view_set_bottom_margin ()
gint	gtk_text_view_get_bottom_margin ()
void	gtk_text_view_set_indent ()
gint	gtk_text_view_get_indent ()
void	gtk_text_view_set_tabs ()
PangoTabArray *	gtk_text_view_get_tabs ()
void	gtk_text_view_set_accepts_tab ()
gboolean	gtk_text_view_get_accepts_tab ()
GtkTextAttributes *	gtk_text_view_get_default_attributes ()
gboolean	gtk_text_view_im_context_filter_keypress ()
void	gtk_text_view_reset_im_context ()
void	gtk_text_view_set_input_purpose ()
GtkInputPurpose	gtk_text_view_get_input_purpose ()
void	gtk_text_view_set_input_hints ()
GtkInputHints	gtk_text_view_get_input_hints ()
void	gtk_text_view_set_monospace ()
gboolean	gtk_text_view_get_monospace ()

Properties

gboolean	accepts-tab	Read / Write
gint	bottom-margin	Read / Write
GtkTextBuffer *	buffer	Read / Write
gboolean	cursor-visible	Read / Write
gboolean	editable	Read / Write
gchar *	im-module	Read / Write
gint	indent	Read / Write

GtkInputHints	input-hints	Read / Write
GtkInputPurpose	input-purpose	Read / Write
GtkJustification	justification	Read / Write
gint	left-margin	Read / Write
gboolean	monospace	Read / Write
gboolean	overwrite	Read / Write
gint	pixels-above-lines	Read / Write
gint	pixels-below-lines	Read / Write
gint	pixels-inside-wrap	Read / Write
gboolean	populate-all	Read / Write
gint	right-margin	Read / Write
PangoTabArray *	tabs	Read / Write
gint	top-margin	Read / Write
GtkWrapMode	wrap-mode	Read / Write

Style Properties

GdkColor *	error-underline-color	Read

Signals

void	backspace	Action
void	copy-clipboard	Action
void	cut-clipboard	Action
void	delete-from-cursor	Action
gboolean	extend-selection	Run Last
void	insert-at-cursor	Action
void	insert-emoji	Action
void	move-cursor	Action
void	move-viewport	Action
void	paste-clipboard	Action
void	populate-popup	Run Last
void	preedit-changed	Action
void	select-all	Action
void	set-anchor	Action
void	toggle-cursor-visible	Action
void	toggle-overwrite	Action

Types and Values

struct	GtkTextView
struct	GtkTextViewClass
enum	GtkTextViewLayer
enum	GtkTextWindowType
enum	GtkTextExtendSelection
enum	GtkWrapMode
struct	GtkTextChildAnchor
#define	GTK_TEXT_VIEW_PRIORITY_VALIDATE

Object Hierarchy

```
GObject
├── GInitiallyUnowned
│   └── GtkWidget
│       └── GtkContainer
│           └── GtkTextView
└── GtkTextChildAnchor
```

Implemented Interfaces

GtkTextView implements AtkImplementorIface, GtkBuildable and GtkScrollable.

Includes

```
#include <gtk/gtk.h>
```

Description

You may wish to begin by reading the text widget conceptual overview which gives an overview of all the objects and data types related to the text widget and how they work together.

CSS nodes

```
textview.view
├── border.top
├── border.left
├── text
│   └── [selection]
├── border.right
├── border.bottom
└── [window.popup]
```

GtkTextView has a main css node with name textview and style class .view, and subnodes for each of the border windows, and the main text area, with names border and text, respectively. The border nodes each get one of the style classes .left, .right, .top or .bottom.

A node representing the selection will appear below the text node.

If a context menu is opened, the window node will appear as a subnode of the main node.

Functions

gtk_text_view_new ()

```
GtkWidget *
gtk_text_view_new (void);
```

Creates a new GtkTextView. If you don't call `gtk_text_view_set_buffer()` before using the text view, an empty default buffer will be created for you. Get the buffer with `gtk_text_view_get_buffer()`. If you want to specify your own buffer, consider `gtk_text_view_new_with_buffer()`.

Returns

a new GtkTextView

---

gtk_text_view_new_with_buffer ()

```
GtkWidget *
gtk_text_view_new_with_buffer (GtkTextBuffer *buffer);
```

Creates a new GtkTextView widget displaying the buffer `buffer` . One buffer can be shared among many widgets. `buffer` may be `NULL` to create a default buffer, in which case this function is equivalent to `gtk_text_view_new()`. The text view adds its own reference count to the buffer; it does not take over an existing reference.

Parameters

buffer	a GtkTextBuffer	

Returns

a new GtkTextView.

---

gtk_text_view_set_buffer ()

```
void
gtk_text_view_set_buffer (GtkTextView *text_view,
                          GtkTextBuffer *buffer);
```

Sets `buffer` as the buffer being displayed by `text_view` . The previous buffer displayed by the text view is unreferenced, and a reference is added to `buffer` . If you owned a reference to `buffer` before passing it to this function, you must remove that reference yourself; GtkTextView will not “adopt” it.

Parameters

text_view	a GtkTextView	
buffer	a GtkTextBuffer.	[allow-none]

---

gtk_text_view_get_buffer ()

```
GtkTextBuffer *
gtk_text_view_get_buffer (GtkTextView *text_view);
```

Returns the GtkTextBuffer being displayed by this text view. The reference count on the buffer is not incremented; the caller of this function won’t own a new reference.

Parameters

text_view	a GtkTextView

Returns

a GtkTextBuffer.

[transfer none]

---

gtk_text_view_get_hadjustment ()

```
GtkAdjustment *
gtk_text_view_get_hadjustment (GtkTextView *text_view);
```

`gtk_text_view_get_hadjustment` has been deprecated since version 3.0 and should not be used in newly-written code.

Use `gtk_scrollable_get_hadjustment()`

Gets the horizontal-scrolling GtkAdjustment.

Parameters

text_view	a GtkTextView

Returns

pointer to the horizontal GtkAdjustment.

[transfer none]

Since: 2.22

---

gtk_text_view_get_vadjustment ()

```
GtkAdjustment *
gtk_text_view_get_vadjustment (GtkTextView *text_view);
```

`gtk_text_view_get_vadjustment` has been deprecated since version 3.0 and should not be used in newly-written code.

Use `gtk_scrollable_get_vadjustment()`

Gets the vertical-scrolling GtkAdjustment.

Parameters

text_view	a GtkTextView

Returns

pointer to the vertical GtkAdjustment.

[transfer none]

Since: 2.22

---

gtk_text_view_scroll_to_mark ()

```
void
gtk_text_view_scroll_to_mark (GtkTextView *text_view,
                              GtkTextMark *mark,
                              gdouble within_margin,
                              gboolean use_align,
                              gdouble xalign,
                              gdouble yalign);
```

Scrolls `text_view` so that `mark` is on the screen in the position indicated by `xalign` and `yalign` . An alignment of 0.0 indicates left or top, 1.0 indicates right or bottom, 0.5 means center. If `use_align` is `FALSE`, the text scrolls the minimal distance to get the mark onscreen, possibly not scrolling at all. The effective screen for purposes of this function is reduced by a margin of size `within_margin` .

Parameters

text_view	a GtkTextView
mark	a GtkTextMark
within_margin	margin as a [0.0,0.5) fraction of screen size
use_align	whether to use alignment arguments (if `FALSE`, just get the mark onscreen)
xalign	horizontal alignment of mark within visible area
yalign	vertical alignment of mark within visible area

---

gtk_text_view_scroll_to_iter ()

```
gboolean
gtk_text_view_scroll_to_iter (GtkTextView *text_view,
                              GtkTextIter *iter,
                              gdouble within_margin,
                              gboolean use_align,
                              gdouble xalign,
                              gdouble yalign);
```

Scrolls `text_view` so that `iter` is on the screen in the position indicated by `xalign` and `yalign` . An alignment of 0.0 indicates left or top, 1.0 indicates right or bottom, 0.5 means center. If `use_align` is `FALSE`, the text scrolls the minimal distance to get the mark onscreen, possibly not scrolling at all. The effective screen for purposes of this function is reduced by a margin of size `within_margin` .

Note that this function uses the currently-computed height of the lines in the text buffer. Line heights are

computed in an idle handler; so this function may not have the desired effect if it's called before the height computations. To avoid oddness, consider using `gtk_text_view_scroll_to_mark()` which saves a point to be scrolled to after line validation.

Parameters

text_view	a GtkTextView
iter	a GtkTextIter
within_margin	margin as a [0.0,0.5) fraction of screen size
use_align	whether to use alignment arguments (if `FALSE`, just get the mark onscreen)
xalign	horizontal alignment of mark within visible area
yalign	vertical alignment of mark within visible area

Returns

`TRUE` if scrolling occurred

---

gtk_text_view_scroll_mark_onscreen ()

```
void
gtk_text_view_scroll_mark_onscreen (GtkTextView *text_view,
                                    GtkTextMark *mark);
```

Scrolls `text_view` the minimum distance such that `mark` is contained within the visible area of the widget.

Parameters

text_view	a GtkTextView
mark	a mark in the buffer for `text_view`

---

gtk_text_view_move_mark_onscreen ()

```
gboolean
gtk_text_view_move_mark_onscreen (GtkTextView *text_view,
                                  GtkTextMark *mark);
```

Moves a mark within the buffer so that it's located within the currently-visible text area.

Parameters

text_view	a GtkTextView
mark	a GtkTextMark

Returns

TRUE if the mark moved (wasn't already onscreen)

---

gtk_text_view_place_cursor_onscreen ()

```
gboolean
gtk_text_view_place_cursor_onscreen (GtkTextView *text_view);
```

Moves the cursor to the currently visible region of the buffer, it it isn't there already.

Parameters

text_view	a GtkTextView	

Returns

TRUE if the cursor had to be moved.

---

gtk_text_view_get_visible_rect ()

```
void
gtk_text_view_get_visible_rect (GtkTextView *text_view,
                                GdkRectangle *visible_rect);
```

Fills `visible_rect` with the currently-visible region of the buffer, in buffer coordinates. Convert to window coordinates with `gtk_text_view_buffer_to_window_coords()`.

Parameters

text_view	a GtkTextView	
visible_rect	rectangle to fill.	[out]

---

gtk_text_view_get_iter_location ()

```
void
gtk_text_view_get_iter_location (GtkTextView *text_view,
                                 const GtkTextIter *iter,
                                 GdkRectangle *location);
```

Gets a rectangle which roughly contains the character at `iter` . The rectangle position is in buffer coordinates; use `gtk_text_view_buffer_to_window_coords()` to convert these coordinates to coordinates for one of the windows in the text view.

Parameters

text_view	a GtkTextView	
iter	a GtkTextIter	
location	bounds of the character at `iter`.	[out]

---

gtk_text_view_get_cursor_locations ()

```
void
gtk_text_view_get_cursor_locations (GtkTextView *text_view,
                                    const GtkTextIter *iter,
                                    GdkRectangle *strong,
                                    GdkRectangle *weak);
```

Given an `iter` within a text layout, determine the positions of the strong and weak cursors if the insertion point is at that iterator. The position of each cursor is stored as a zero-width rectangle. The strong cursor location is the location where characters of the directionality equal to the base direction of the paragraph are inserted. The weak cursor location is the location where characters of the directionality opposite to the base direction of the paragraph are inserted.

If `iter` is `NULL`, the actual cursor position is used.

Note that if `iter` happens to be the actual cursor position, and there is currently an IM preedit sequence being entered, the returned locations will be adjusted to account for the preedit cursor's offset within the preedit sequence.

The rectangle position is in buffer coordinates; use `gtk_text_view_buffer_to_window_coords()` to convert these coordinates to coordinates for one of the windows in the text view.

Parameters

text_view	a GtkTextView	
iter	a GtkTextIter.	[allow-none]
strong	location to store the strong cursor position (may be `NULL`).	[out][allow-none]
weak	location to store the weak cursor position (may be `NULL`).	[out][allow-none]

Since: 3.0

---

gtk_text_view_get_line_at_y ()

```
void
gtk_text_view_get_line_at_y (GtkTextView *text_view,
                             GtkTextIter *target_iter,
                             gint y,
                             gint *line_top);
```

Gets the GtkTextIter at the start of the line containing the coordinate `y`. `y` is in buffer coordinates, convert from window coordinates with `gtk_text_view_window_to_buffer_coords()`. If non-`NULL`, `line_top` will be filled with the coordinate of the top edge of the line.

Parameters

text_view	a GtkTextView	
target_iter	a GtkTextIter.	[out]
y	a y coordinate	
line_top	return location for top coordinate of the line.	[out]

---

gtk_text_view_get_line_yrange ()

```
void
gtk_text_view_get_line_yrange (GtkTextView *text_view,
                               const GtkTextIter *iter,
                               gint *y,
                               gint *height);
```

Gets the y coordinate of the top of the line containing `iter` , and the height of the line. The coordinate is a buffer coordinate; convert to window coordinates with `gtk_text_view_buffer_to_window_coords()`.

Parameters

text_view	a GtkTextView	
iter	a GtkTextIter	
y	return location for a y coordinate.	[out]
height	return location for a height.	[out]

---

gtk_text_view_get_iter_at_location ()

```
gboolean
gtk_text_view_get_iter_at_location (GtkTextView *text_view,
                                    GtkTextIter *iter,
                                    gint x,
                                    gint y);
```

Retrieves the iterator at buffer coordinates `x` and `y` . Buffer coordinates are coordinates for the entire buffer, not just the currently-displayed portion. If you have coordinates from an event, you have to convert those to buffer coordinates with `gtk_text_view_window_to_buffer_coords()`.

Parameters

text_view	a GtkTextView	
iter	a GtkTextIter.	[out]
x	x position, in buffer coordinates	
y	y position, in buffer coordinates	

Returns

`TRUE` if the position is over text

---

gtk_text_view_get_iter_at_position ()

```
gboolean
gtk_text_view_get_iter_at_position (GtkTextView *text_view,
                                    GtkTextIter *iter,
                                    gint *trailing,
                                    gint x,
                                    gint y);
```

Retrieves the iterator pointing to the character at buffer coordinates `x` and `y` . Buffer coordinates are coordinates for the entire buffer, not just the currently-displayed portion. If you have coordinates from an event, you have to convert those to buffer coordinates with `gtk_text_view_window_to_buffer_coords()`.

Note that this is different from `gtk_text_view_get_iter_at_location()`, which returns cursor locations, i.e. positions between characters.

Parameters

text_view	a GtkTextView	
iter	a GtkTextIter.	[out]
trailing	if non-`NULL`, location to store an integer indicating where in the grapheme the user clicked. It will either be zero, or the number of characters in the grapheme. 0 represents the trailing edge of the grapheme.	[out][allow-none]
x	x position, in buffer coordinates	
y	y position, in buffer coordinates	

Returns

`TRUE` if the position is over text

Since: 2.6

---

gtk_text_view_buffer_to_window_coords ()

```
void
gtk_text_view_buffer_to_window_coords (GtkTextView *text_view,
                                       GtkTextWindowType win,
                                       gint buffer_x,
                                       gint buffer_y,
                                       gint *window_x,
                                       gint *window_y);
```

Converts coordinate (`buffer_x` , `buffer_y`) to coordinates for the window `win` , and stores the result in (`window_x` , `window_y`).

Note that you can’t convert coordinates for a nonexisting window (see `gtk_text_view_set_border_window_size()`).

Parameters

text_view	a GtkTextView	
win	a GtkTextWindowType, except GTK_TEXT_WINDOW_PRIVATE	
buffer_x	buffer x coordinate	
buffer_y	buffer y coordinate	
window_x	window x coordinate return location or NULL.	[out][allow-none]
window_y	window y coordinate return location or NULL.	[out][allow-none]

---

gtk_text_view_window_to_buffer_coords ()

```
void
gtk_text_view_window_to_buffer_coords (GtkTextView *text_view,
                                       GtkTextWindowType win,
                                       gint window_x,
                                       gint window_y,
                                       gint *buffer_x,
                                       gint *buffer_y);
```

Converts coordinates on the window identified by win to buffer coordinates, storing the result in (buffer_x ,buffer_y).

Note that you can't convert coordinates for a nonexisting window (see gtk_text_view_set_border_window_size()).

Parameters

text_view	a GtkTextView	
win	a GtkTextWindowType except GTK_TEXT_WINDOW_PRIVATE	
window_x	window x coordinate	
window_y	window y coordinate	
buffer_x	buffer x coordinate return location or NULL.	[out][allow-none]
buffer_y	buffer y coordinate return location or NULL.	[out][allow-none]

---

gtk_text_view_get_window ()

```
GdkWindow *
gtk_text_view_get_window (GtkTextView *text_view,
                          GtkTextWindowType win);
```

Retrieves the GdkWindow corresponding to an area of the text view; possible windows include the overall widget window, child windows on the left, right, top, bottom, and the window that displays the text buffer. Windows are NULL and nonexistent if their width or height is 0, and are nonexistent before the widget has been realized.

Parameters

text_view	a GtkTextView
win	window to get

Returns

a GdkWindow, or `NULL`.

[nullable][transfer none]

---

gtk_text_view_get_window_type ()

```
GtkTextWindowType
gtk_text_view_get_window_type (GtkTextView *text_view,
                               GdkWindow *window);
```

Usually used to find out which window an event corresponds to.

If you connect to an event signal on `text_view` , this function should be called on `event->window` to see which window it was.

Parameters

text_view	a GtkTextView
window	a window type

Returns

the window type.

---

gtk_text_view_set_border_window_size ()

```
void
gtk_text_view_set_border_window_size (GtkTextView *text_view,
                                      GtkTextWindowType type,
                                      gint size);
```

Sets the width of `GTK_TEXT_WINDOW_LEFT` or `GTK_TEXT_WINDOW_RIGHT`, or the height of `GTK_TEXT_WINDOW_TOP` or `GTK_TEXT_WINDOW_BOTTOM`. Automatically destroys the corresponding window if the size is set to 0, and creates the window if the size is set to non-zero. This function can only be used for the "border windows", and it won't work with `GTK_TEXT_WINDOW_WIDGET`, `GTK_TEXT_WINDOW_TEXT`, or `GTK_TEXT_WINDOW_PRIVATE`.

Parameters

text_view	a GtkTextView
type	window to affect
size	width or height of the window

---

gtk_text_view_get_border_window_size ()

```
gint
gtk_text_view_get_border_window_size (GtkTextView *text_view,
                                      GtkTextWindowType type);
```

Gets the width of the specified border window. See `gtk_text_view_set_border_window_size()`.

Parameters

text_view	a GtkTextView
type	window to return size from

Returns

width of window

---

gtk_text_view_forward_display_line ()

```
gboolean
gtk_text_view_forward_display_line (GtkTextView *text_view,
                                    GtkTextIter *iter);
```

Moves the given `iter` forward by one display (wrapped) line. A display line is different from a paragraph. Paragraphs are separated by newlines or other paragraph separator characters. Display lines are created by line-wrapping a paragraph. If wrapping is turned off, display lines and paragraphs will be the same. Display lines are divided differently for each view, since they depend on the view's width; paragraphs are the same in all views, since they depend on the contents of the GtkTextBuffer.

Parameters

text_view	a GtkTextView
iter	a GtkTextIter

Returns

`TRUE` if `iter` was moved and is not on the end iterator

---

gtk_text_view_backward_display_line ()

```
gboolean
gtk_text_view_backward_display_line (GtkTextView *text_view,
                                     GtkTextIter *iter);
```

Moves the given `iter` backward by one display (wrapped) line. A display line is different from a paragraph. Paragraphs are separated by newlines or other paragraph separator characters. Display lines are created by line-wrapping a paragraph. If wrapping is turned off, display lines and paragraphs will be the same. Display lines are divided differently for each view, since they depend on the view's width; paragraphs are the same in all views,

since they depend on the contents of the GtkTextBuffer.

Parameters

text_view	a GtkTextView
iter	a GtkTextIter

Returns

TRUE if iter was moved and is not on the end iterator

---

gtk_text_view_forward_display_line_end ()

```
gboolean
gtk_text_view_forward_display_line_end
                                (GtkTextView *text_view,
                                 GtkTextIter *iter);
```

Moves the given iter forward to the next display line end. A display line is different from a paragraph. Paragraphs are separated by newlines or other paragraph separator characters. Display lines are created by line-wrapping a paragraph. If wrapping is turned off, display lines and paragraphs will be the same. Display lines are divided differently for each view, since they depend on the view's width; paragraphs are the same in all views, since they depend on the contents of the GtkTextBuffer.

Parameters

text_view	a GtkTextView
iter	a GtkTextIter

Returns

TRUE if iter was moved and is not on the end iterator

---

gtk_text_view_backward_display_line_start ()

```
gboolean
gtk_text_view_backward_display_line_start
                                (GtkTextView *text_view,
                                 GtkTextIter *iter);
```

Moves the given iter backward to the next display line start. A display line is different from a paragraph. Paragraphs are separated by newlines or other paragraph separator characters. Display lines are created by line-wrapping a paragraph. If wrapping is turned off, display lines and paragraphs will be the same. Display lines are divided differently for each view, since they depend on the view's width; paragraphs are the same in all views, since they depend on the contents of the GtkTextBuffer.

Parameters

text_view	a GtkTextView
iter	a GtkTextIter

Returns

`TRUE` if `iter` was moved and is not on the end iterator

---

gtk_text_view_starts_display_line ()

```
gboolean
gtk_text_view_starts_display_line (GtkTextView *text_view,
                                   const GtkTextIter *iter);
```

Determines whether `iter` is at the start of a display line. See `gtk_text_view_forward_display_line()` for an explanation of display lines vs. paragraphs.

Parameters

text_view	a GtkTextView
iter	a GtkTextIter

Returns

`TRUE` if `iter` begins a wrapped line

---

gtk_text_view_move_visually ()

```
gboolean
gtk_text_view_move_visually (GtkTextView *text_view,
                             GtkTextIter *iter,
                             gint count);
```

Move the iterator a given number of characters visually, treating it as the strong cursor position. If `count` is positive, then the new strong cursor position will be `count` positions to the right of the old cursor position. If `count` is negative then the new strong cursor position will be `count` positions to the left of the old cursor position.

In the presence of bi-directional text, the correspondence between logical and visual order will depend on the direction of the current run, and there may be jumps when the cursor is moved off of the end of a run.

Parameters

text_view	a GtkTextView
iter	a GtkTextIter
count	number of characters to move (negative moves left, positive moves right)

Returns

`TRUE` if `iter` moved and is not on the end iterator

---

gtk_text_view_add_child_at_anchor ()

```
void
gtk_text_view_add_child_at_anchor (GtkTextView *text_view,
                                   GtkWidget *child,
                                   GtkTextChildAnchor *anchor);
```

Adds a child widget in the text buffer, at the given `anchor` .

Parameters

text_view	a GtkTextView
child	a GtkWidget
anchor	a GtkTextChildAnchor in the GtkTextBuffer for `text_view`

---

gtk_text_child_anchor_new ()

```
GtkTextChildAnchor *
gtk_text_child_anchor_new (void);
```

Creates a new GtkTextChildAnchor. Usually you would then insert it into a GtkTextBuffer with `gtk_text_buffer_insert_child_anchor()`. To perform the creation and insertion in one step, use the convenience function `gtk_text_buffer_create_child_anchor()`.

Returns

a new GtkTextChildAnchor

---

gtk_text_child_anchor_get_widgets ()

```
GList *
gtk_text_child_anchor_get_widgets (GtkTextChildAnchor *anchor);
```

Gets a list of all widgets anchored at this child anchor. The returned list should be freed with `g_list_free()`.

Parameters

anchor	a GtkTextChildAnchor

Returns

list of widgets anchored at `anchor` .

[element-type GtkWidget][transfer container]

---

gtk_text_child_anchor_get_deleted ()

```
gboolean
gtk_text_child_anchor_get_deleted (GtkTextChildAnchor *anchor);
```

Determines whether a child anchor has been deleted from the buffer. Keep in mind that the child anchor will be unreferenced when removed from the buffer, so you need to hold your own reference (with `g_object_ref()`) if you plan to use this function — otherwise all deleted child anchors will also be finalized.

Parameters

anchor	a GtkTextChildAnchor

Returns

`TRUE` if the child anchor has been deleted from its buffer

---

gtk_text_view_add_child_in_window ()

```
void
gtk_text_view_add_child_in_window (GtkTextView *text_view,
                                   GtkWidget *child,
                                   GtkTextWindowType which_window,
                                   gint xpos,
                                   gint ypos);
```

Adds a child at fixed coordinates in one of the text widget's windows.

The window must have nonzero size (see `gtk_text_view_set_border_window_size()`). Note that the child coordinates are given relative to scrolling. When placing a child in GTK_TEXT_WINDOW_WIDGET, scrolling is irrelevant, the child floats above all scrollable areas. But when placing a child in one of the scrollable windows (border windows or text window) it will move with the scrolling as needed.

Parameters

text_view	a GtkTextView
child	a GtkWidget
which_window	which window the child should appear in
xpos	X position of child in window coordinates
ypos	Y position of child in window coordinates

---

gtk_text_view_move_child ()

```
void
gtk_text_view_move_child (GtkTextView *text_view,
                          GtkWidget *child,
                          gint xpos,
                          gint ypos);
```

Updates the position of a child, as for `gtk_text_view_add_child_in_window()`.

Parameters

text_view	a GtkTextView
child	child widget already added to the text view
xpos	new X position in window coordinates
ypos	new Y position in window coordinates

---

gtk_text_view_set_wrap_mode ()

```
void
gtk_text_view_set_wrap_mode (GtkTextView *text_view,
                             GtkWrapMode wrap_mode);
```

Sets the line wrapping for the view.

Parameters

text_view	a GtkTextView
wrap_mode	a GtkWrapMode

---

gtk_text_view_get_wrap_mode ()

```
GtkWrapMode
gtk_text_view_get_wrap_mode (GtkTextView *text_view);
```

Gets the line wrapping for the view.

Parameters

text_view	a GtkTextView

Returns

the line wrap setting

---

gtk_text_view_set_editable ()

```
void
gtk_text_view_set_editable (GtkTextView *text_view,
                            gboolean setting);
```

Sets the default editability of the GtkTextView. You can override this default setting with tags in the buffer, using the “editable” attribute of tags.

Parameters

text_view	a GtkTextView
setting	whether it’s editable

---

gtk_text_view_get_editable ()

```
gboolean
gtk_text_view_get_editable (GtkTextView *text_view);
```

Returns the default editability of the GtkTextView. Tags in the buffer may override this setting for some ranges of text.

Parameters

text_view	a GtkTextView

Returns

whether text is editable by default

---

gtk_text_view_set_cursor_visible ()

```
void
gtk_text_view_set_cursor_visible (GtkTextView *text_view,
                                  gboolean setting);
```

Toggles whether the insertion point should be displayed. A buffer with no editable text probably shouldn’t have a visible cursor, so you may want to turn the cursor off.

Note that this property may be overridden by the “gtk-keynave-use-caret” settings.

Parameters

text_view	a GtkTextView
setting	whether to show the insertion cursor

---

gtk_text_view_get_cursor_visible ()

```
gboolean
gtk_text_view_get_cursor_visible (GtkTextView *text_view);
```

Find out whether the cursor should be displayed.

Parameters

text_view	a GtkTextView

Returns

whether the insertion mark is visible

---

gtk_text_view_reset_cursor_blink ()

```
void
gtk_text_view_reset_cursor_blink (GtkTextView *text_view);
```

Ensures that the cursor is shown (i.e. not in an 'off' blink interval) and resets the time that it will stay blinking (or visible, in case blinking is disabled).

This function should be called in response to user input (e.g. from derived classes that override the textview's "key-press-event" handler).

Parameters

text_view	a GtkTextView

Since: 3.20

---

gtk_text_view_set_overwrite ()

```
void
gtk_text_view_set_overwrite (GtkTextView *text_view,
                             gboolean overwrite);
```

Changes the GtkTextView overwrite mode.

Parameters

text_view	a GtkTextView
overwrite	TRUE to turn on overwrite mode, FALSE to turn it off

Since: 2.4

---

gtk_text_view_get_overwrite ()

```
gboolean
gtk_text_view_get_overwrite (GtkTextView *text_view);
```

Returns whether the GtkTextView is in overwrite mode or not.

Parameters

text_view	a GtkTextView

Returns

whether `text_view` is in overwrite mode or not.

Since: 2.4

---

gtk_text_view_set_pixels_above_lines ()

```
void
gtk_text_view_set_pixels_above_lines (GtkTextView *text_view,
                                      gint pixels_above_lines);
```

Sets the default number of blank pixels above paragraphs in `text_view` . Tags in the buffer for `text_view` may override the defaults.

Parameters

text_view	a GtkTextView
pixels_above_lines	pixels above paragraphs

---

gtk_text_view_get_pixels_above_lines ()

```
gint
gtk_text_view_get_pixels_above_lines (GtkTextView *text_view);
```

Gets the default number of pixels to put above paragraphs. Adding this function with `gtk_text_view_get_pixels_below_lines()` is equal to the line space between each paragraph.

Parameters

text_view	a GtkTextView

Returns

default number of pixels above paragraphs

---

gtk_text_view_set_pixels_below_lines ()

```
void
gtk_text_view_set_pixels_below_lines (GtkTextView *text_view,
                                      gint pixels_below_lines);
```

Sets the default number of pixels of blank space to put below paragraphs in `text_view` . May be overridden by tags applied to `text_view` 's buffer.

Parameters

text_view	a GtkTextView
pixels_below_lines	pixels below paragraphs

---

gtk_text_view_get_pixels_below_lines ()

```
gint
gtk_text_view_get_pixels_below_lines (GtkTextView *text_view);
```

Gets the value set by `gtk_text_view_set_pixels_below_lines()`.

The line space is the sum of the value returned by this function and the value returned by `gtk_text_view_get_pixels_above_lines()`.

Parameters

text_view	a GtkTextView

Returns

default number of blank pixels below paragraphs

---

gtk_text_view_set_pixels_inside_wrap ()

```
void
gtk_text_view_set_pixels_inside_wrap (GtkTextView *text_view,
                                      gint pixels_inside_wrap);
```

Sets the default number of pixels of blank space to leave between display/wrapped lines within a paragraph. May be overridden by tags in `text_view` 's buffer.

Parameters

text_view	a GtkTextView
pixels_inside_wrap	default number of pixels between wrapped lines

---

gtk_text_view_get_pixels_inside_wrap ()

```
gint
gtk_text_view_get_pixels_inside_wrap (GtkTextView *text_view);
```

Gets the value set by `gtk_text_view_set_pixels_inside_wrap()`.

Parameters

text_view	a GtkTextView

Returns

default number of pixels of blank space between wrapped lines

---

gtk_text_view_set_justification ()

```
void
gtk_text_view_set_justification (GtkTextView *text_view,
                                 GtkJustification justification);
```

Sets the default justification of text in `text_view` . Tags in the view's buffer may override the default.

Parameters

text_view	a GtkTextView
justification	justification

---

gtk_text_view_get_justification ()

```
GtkJustification
gtk_text_view_get_justification (GtkTextView *text_view);
```

Gets the default justification of paragraphs in `text_view` . Tags in the buffer may override the default.

Parameters

text_view	a GtkTextView

Returns

default justification

---

gtk_text_view_set_left_margin ()

```
void
gtk_text_view_set_left_margin (GtkTextView *text_view,
```

```
                                gint left_margin);
```

Sets the default left margin for text in `text_view` . Tags in the buffer may override the default.

Note that this function is confusingly named. In CSS terms, the value set here is padding.

Parameters

text_view	a GtkTextView
left_margin	left margin in pixels

---

gtk_text_view_get_left_margin ()

```
gint
gtk_text_view_get_left_margin (GtkTextView *text_view);
```

Gets the default left margin size of paragraphs in the `text_view` . Tags in the buffer may override the default.

Parameters

text_view	a GtkTextView

Returns

left margin in pixels

---

gtk_text_view_set_right_margin ()

```
void
gtk_text_view_set_right_margin (GtkTextView *text_view,
                                gint right_margin);
```

Sets the default right margin for text in the text view. Tags in the buffer may override the default.

Note that this function is confusingly named. In CSS terms, the value set here is padding.

Parameters

text_view	a GtkTextView
right_margin	right margin in pixels

---

gtk_text_view_get_right_margin ()

```
gint
gtk_text_view_get_right_margin (GtkTextView *text_view);
```

Gets the default right margin for text in `text_view` . Tags in the buffer may override the default.

Parameters

text_view	a GtkTextView

Returns

right margin in pixels

---

gtk_text_view_set_top_margin ()

```
void
gtk_text_view_set_top_margin (GtkTextView *text_view,
                              gint top_margin);
```

Sets the top margin for text in `text_view`.

Note that this function is confusingly named. In CSS terms, the value set here is padding.

Parameters

text_view	a GtkTextView
top_margin	top margin in pixels

Since: 3.18

---

gtk_text_view_get_top_margin ()

```
gint
gtk_text_view_get_top_margin (GtkTextView *text_view);
```

Gets the top margin for text in the `text_view`.

Parameters

text_view	a GtkTextView

Returns

top margin in pixels

Since: 3.18

---

gtk_text_view_set_bottom_margin ()

```
void
gtk_text_view_set_bottom_margin (GtkTextView *text_view,
                                 gint bottom_margin);
```

Sets the bottom margin for text in `text_view`.

Note that this function is confusingly named. In CSS terms, the value set here is padding.

Parameters

text_view	a GtkTextView
bottom_margin	bottom margin in pixels

Since: 3.18

---

gtk_text_view_get_bottom_margin ()

```
gint
gtk_text_view_get_bottom_margin (GtkTextView *text_view);
```

Gets the bottom margin for text in the `text_view` .

Parameters

text_view	a GtkTextView

Returns

bottom margin in pixels

Since: 3.18

---

gtk_text_view_set_indent ()

```
void
gtk_text_view_set_indent (GtkTextView *text_view,
                          gint indent);
```

Sets the default indentation for paragraphs in `text_view` . Tags in the buffer may override the default.

Parameters

text_view	a GtkTextView
indent	indentation in pixels

---

gtk_text_view_get_indent ()

```
gint
gtk_text_view_get_indent (GtkTextView *text_view);
```

Gets the default indentation of paragraphs in `text_view` . Tags in the view’s buffer may override the default. The indentation may be negative.

Parameters

text_view	a GtkTextView

Returns

number of pixels of indentation

---

gtk_text_view_set_tabs ()

```
void
gtk_text_view_set_tabs (GtkTextView *text_view,
                        PangoTabArray *tabs);
```

Sets the default tab stops for paragraphs in `text_view` . Tags in the buffer may override the default.

Parameters

text_view	a GtkTextView
tabs	tabs as a PangoTabArray

---

gtk_text_view_get_tabs ()

```
PangoTabArray *
gtk_text_view_get_tabs (GtkTextView *text_view);
```

Gets the default tabs for `text_view` . Tags in the buffer may override the defaults. The returned array will be `NULL` if “standard” (8-space) tabs are used. Free the return value with `pango_tab_array_free()`.

Parameters

text_view	a GtkTextView

Returns

copy of default tab array, or `NULL` if “standard" tabs are used; must be freed with `pango_tab_array_free()`.

[nullable][transfer full]

---

gtk_text_view_set_accepts_tab ()

```
void
gtk_text_view_set_accepts_tab (GtkTextView *text_view,
                               gboolean accepts_tab);
```

Sets the behavior of the text widget when the Tab key is pressed. If `accepts_tab` is `TRUE`, a tab character is inserted. If `accepts_tab` is `FALSE` the keyboard focus is moved to the next widget in the focus chain.

Parameters

text_view	A GtkTextView
accepts_tab	TRUE if pressing the Tab key should insert a tab character, FALSE, if pressing the Tab key should move the keyboard focus.

Since: 2.4

---

gtk_text_view_get_accepts_tab ()

```
gboolean
gtk_text_view_get_accepts_tab (GtkTextView *text_view);
```

Returns whether pressing the Tab key inserts a tab characters. gtk_text_view_set_accepts_tab().

Parameters

text_view	A GtkTextView

Returns

TRUE if pressing the Tab key inserts a tab character, FALSE if pressing the Tab key moves the keyboard focus.

Since: 2.4

---

gtk_text_view_get_default_attributes ()

```
GtkTextAttributes *
gtk_text_view_get_default_attributes (GtkTextView *text_view);
```

Obtains a copy of the default text attributes. These are the attributes used for text unless a tag overrides them. You'd typically pass the default attributes in to gtk_text_iter_get_attributes() in order to get the attributes in effect at a given text position.

The return value is a copy owned by the caller of this function, and should be freed with gtk_text_attributes_unref().

Parameters

text_view	a GtkTextView

Returns

a new GtkTextAttributes

---

gtk_text_view_im_context_filter_keypress ()

```
gboolean
gtk_text_view_im_context_filter_keypress
                                (GtkTextView *text_view,
                                 GdkEventKey *event);
```

Allow the GtkTextView input method to internally handle key press and release events. If this function returns `TRUE`, then no further processing should be done for this key event. See `gtk_im_context_filter_keypress()`.

Note that you are expected to call this function from your handler when overriding key event handling. This is needed in the case when you need to insert your own key handling between the input method and the default key event handling of the GtkTextView.

```
static gboolean
gtk_foo_bar_key_press_event (GtkWidget
*widget,
                             GdkEventKey
*event)
{
  guint keyval;

  gdk_event_get_keyval ((GdkEvent*)event,
&keyval);

  if (keyval == GDK_KEY_Return || keyval ==
GDK_KEY_KP_Enter)
    {
      if
(gtk_text_view_im_context_filter_keypress
(GTK_TEXT_VIEW (widget), event))
        return TRUE;
    }

  // Do some stuff

  return GTK_WIDGET_CLASS
(gtk_foo_bar_parent_class)->key_press_event
(widget, event);
}
```

Parameters

text_view	a GtkTextView
event	the key event

Returns

`TRUE` if the input method handled the key event.

Since: 2.22

---

gtk_text_view_reset_im_context ()

```
void
gtk_text_view_reset_im_context (GtkTextView *text_view);
```

Reset the input method context of the text view if needed.

This can be necessary in the case where modifying the buffer would confuse on-going input method behavior.

Parameters

text_view	a GtkTextView

Since: 2.22

---

gtk_text_view_set_input_purpose ()

```
void
gtk_text_view_set_input_purpose (GtkTextView *text_view,
                                 GtkInputPurpose purpose);
```

Sets the “input-purpose” property which can be used by on-screen keyboards and other input methods to adjust their behaviour.

Parameters

text_view	a GtkTextView
purpose	the purpose

Since: 3.6

---

gtk_text_view_get_input_purpose ()

```
GtkInputPurpose
gtk_text_view_get_input_purpose (GtkTextView *text_view);
```

Gets the value of the “input-purpose” property.

Parameters

text_view	a GtkTextView

Since: 3.6

---

gtk_text_view_set_input_hints ()

```
void
gtk_text_view_set_input_hints (GtkTextView *text_view,
                               GtkInputHints hints);
```

Sets the “input-hints” property, which allows input methods to fine-tune their behaviour.

Parameters

text_view	a GtkTextView
hints	the hints

Since: 3.6

---

gtk_text_view_get_input_hints ()

```
GtkInputHints
gtk_text_view_get_input_hints (GtkTextView *text_view);
```

Gets the value of the “input-hints” property.

Parameters

text_view	a GtkTextView

Since: 3.6

---

gtk_text_view_set_monospace ()

```
void
gtk_text_view_set_monospace (GtkTextView *text_view,
                             gboolean monospace);
```

Sets the “monospace” property, which indicates that the text view should use monospace fonts.

Parameters

text_view	a GtkTextView
monospace	TRUE to request monospace styling

Since: 3.16

---

gtk_text_view_get_monospace ()

```
gboolean
gtk_text_view_get_monospace (GtkTextView *text_view);
```

Gets the value of the “monospace” property.

Return: TRUE if monospace fonts are desired

Parameters

text_view	a GtkTextView

Since: 3.16

Types and Values

struct GtkTextView

```
struct GtkTextView;
```

---

struct GtkTextViewClass

```
struct GtkTextViewClass {
  GtkContainerClass parent_class;


  void (* populate_popup)        (GtkTextView      *text_view,
                                  GtkWidget        *popup);
  void (* move_cursor)           (GtkTextView      *text_view,
                                  GtkMovementStep   step,
                                  gint              count,
                                  gboolean          extend_selection);
  void (* set_anchor)            (GtkTextView      *text_view);
  void (* insert_at_cursor)      (GtkTextView      *text_view,
                                  const gchar      *str);
  void (* delete_from_cursor)    (GtkTextView      *text_view,
                                  GtkDeleteType     type,
                                  gint              count);
  void (* backspace)             (GtkTextView      *text_view);
  void (* cut_clipboard)         (GtkTextView      *text_view);
  void (* copy_clipboard)        (GtkTextView      *text_view);
  void (* paste_clipboard)       (GtkTextView      *text_view);
  void (* toggle_overwrite)      (GtkTextView      *text_view);
  GtkTextBuffer * (* create_buffer) (GtkTextView   *text_view);
  void (* draw_layer)            (GtkTextView      *text_view,
                             GtkTextViewLayer  layer,
                             cairo_t          *cr);
  gboolean (* extend_selection)  (GtkTextView            *text_view,
                                  GtkTextExtendSelection  granularity,
                                  const GtkTextIter      *location,
                                  GtkTextIter            *start,
                                  GtkTextIter            *end);
  void (* insert_emoji)          (GtkTextView      *text_view);
};
```

Members

`populate_popup` ()	The class handler for the “populate-popup” signal.
`move_cursor` ()	The class handler for the “move-cursor” keybinding signal.
`set_anchor` ()	The class handler for the “set-anchor” keybinding signal.
`insert_at_cursor` ()	The class handler for the “insert-at-cursor” keybinding signal.
`delete_from_cursor` ()	The class handler for the “delete-from-cursor” keybinding signal.
`backspace` ()	The class handler for the “backspace” keybinding signal.
`cut_clipboard` ()	The class handler for the “cut-clipboard” keybinding signal
`copy_clipboard` ()	The class handler for the “copy-

	clipboard” keybinding signal.
`paste_clipboard` ()	The class handler for the “paste-clipboard” keybinding signal.
`toggle_overwrite` ()	The class handler for the “toggle-overwrite” keybinding signal.
`create_buffer` ()	The create_buffer vfunc is called to create a GtkTextBuffer for the text view. The default implementation is to just call `gtk_text_buffer_new()`. Since: 3.10
`draw_layer` ()	The draw_layer vfunc is called before and after the text view is drawing its own text. Applications can override this vfunc in a subclass to draw customized content underneath or above the text. In the `GTK_TEXT_VIEW_LAYER_BELOW_TEXT` and `GTK_TEXT_VIEW_LAYER_ABOVE_TEXT` the drawing is done in the buffer coordinate space, but the older (deprecated) layers `GTK_TEXT_VIEW_LAYER_BELOW` and `GTK_TEXT_VIEW_LAYER_ABOVE` work in viewport coordinates, which makes them unnecessarily hard to use. Since: 3.14
`extend_selection` ()	The class handler for the “extend-selection” signal. Since 3.16
`insert_emoji` ()	

---

enum GtkTextViewLayer

Used to reference the layers of GtkTextView for the purpose of customized drawing with the ::draw_layer vfunc.

Members

GTK_TEXT_VIEW_LAYER_BELOW	Old deprecated layer, use `GTK_TEXT_VIEW_LAYER_BELOW_TEXT` instead
GTK_TEXT_VIEW_LAYER_ABOVE	Old deprecated layer, use `GTK_TEXT_VIEW_LAYER_ABOVE_TEXT` instead
GTK_TEXT_VIEW_LAYER_BELOW_TEXT	The layer rendered below the text (but above the background). Since: 3.20
GTK_TEXT_VIEW_LAYER_ABO	The layer rendered above the text.

VE_TEXT	Since: 3.20

enum GtkTextWindowType

Used to reference the parts of GtkTextView.

Members

GTK_TEXT_WINDOW_PRIVATE	Invalid value, used as a marker
GTK_TEXT_WINDOW_WIDGET	Window that floats over scrolling areas.
GTK_TEXT_WINDOW_TEXT	Scrollable text window.
GTK_TEXT_WINDOW_LEFT	Left side border window.
GTK_TEXT_WINDOW_RIGHT	Right side border window.
GTK_TEXT_WINDOW_TOP	Top border window.
GTK_TEXT_WINDOW_BOTTOM	Bottom border window.

enum GtkTextExtendSelection

Granularity types that extend the text selection. Use the "extend-selection" signal to customize the selection.

Members

GTK_TEXT_EXTEND_SELECTION_WORD	Selects the current word. It is triggered by a double-click for example.
GTK_TEXT_EXTEND_SELECTION_LINE	Selects the current line. It is triggered by a triple-click for example.

Since: 3.16

enum GtkWrapMode

Describes a type of line wrapping.

Members

GTK_WRAP_NONE	do not wrap lines; just make the text area wider
GTK_WRAP_CHAR	wrap text, breaking lines anywhere the cursor can appear (between characters, usually - if you want to

	be technical, between graphemes, see `pango_get_log_attrs()`)
GTK_WRAP_WORD	wrap text, breaking lines in between words
GTK_WRAP_WORD_CHAR	wrap text, breaking lines in between words, or if that is not enough, also between graphemes

---

struct GtkTextChildAnchor

```
struct GtkTextChildAnchor;
```

A GtkTextChildAnchor is a spot in the buffer where child widgets can be “anchored” (inserted inline, as if they were characters). The anchor can have multiple widgets anchored, to allow for multiple views.

GTK_TEXT_VIEW_PRIORITY_VALIDATE

```
#define GTK_TEXT_VIEW_PRIORITY_VALIDATE (GDK_PRIORITY_REDRAW + 5)
```

The priority at which the text view validates onscreen lines in an idle job in the background.

Property Details

The `“accepts-tab”` property

```
  “accepts-tab”              gboolean
```

Whether Tab will result in a tab character being entered.

Flags: Read / Write

Default value: TRUE

The `“bottom-margin”` property

```
  “bottom-margin”            gint
```

The bottom margin for text in the text view.

Note that this property is confusingly named. In CSS terms, the value set here is padding, and it is applied in addition to the padding from the theme.

Don't confuse this property with “margin-bottom”.

Flags: Read / Write

Allowed values: >= 0

Default value: 0

Since: 3.18

The “buffer” property

“buffer” GtkTextBuffer *

The buffer which is displayed.

Flags: Read / Write

The “cursor-visible” property

“cursor-visible” gboolean

If the insertion cursor is shown.

Flags: Read / Write

Default value: TRUE

The “editable” property

“editable” gboolean

Whether the text can be modified by the user.

Flags: Read / Write

Default value: TRUE

The “im-module” property

“im-module” gchar *

Which IM (input method) module should be used for this text_view. See GtkIMContext.

Setting this to a non-NULL value overrides the system-wide IM module setting. See the GtkSettings “gtk-im-module” property.

Flags: Read / Write

Default value: NULL

Since: 2.16

The “indent” property

“indent” gint

Amount to indent the paragraph, in pixels.

Flags: Read / Write

Default value: 0

The “input-hints” property

“input-hints” GtkInputHints

Additional hints (beyond “input-purpose”) that allow input methods to fine-tune their behaviour.

Flags: Read / Write

Since: 3.6

The “input-purpose” property

“input-purpose” GtkInputPurpose

The purpose of this text field.

This property can be used by on-screen keyboards and other input methods to adjust their behaviour.

Flags: Read / Write

Default value: GTK_INPUT_PURPOSE_FREE_FORM

Since: 3.6

The “justification” property

“justification” GtkJustification

Left, right, or center justification.

Flags: Read / Write

Default value: GTK_JUSTIFY_LEFT

The “left-margin” property

“left-margin” gint

The default left margin for text in the text view. Tags in the buffer may override the default.

Note that this property is confusingly named. In CSS terms, the value set here is padding, and it is applied in addition to the padding from the theme.

Don't confuse this property with “margin-left”.

Flags: Read / Write

Allowed values: >= 0

Default value: 0

The “monospace” property

“monospace” gboolean

Whether to use a monospace font.

Flags: Read / Write

Default value: FALSE

The “overwrite” property

“overwrite” gboolean

Whether entered text overwrites existing contents.

Flags: Read / Write

Default value: FALSE

The “pixels-above-lines” property

“pixels-above-lines” gint

Pixels of blank space above paragraphs.

Flags: Read / Write

Allowed values: >= 0

Default value: 0

The “pixels-below-lines” property

“pixels-below-lines” gint

Pixels of blank space below paragraphs.

Flags: Read / Write

Allowed values: >= 0

Default value: 0

The “pixels-inside-wrap” property

“pixels-inside-wrap” gint

Pixels of blank space between wrapped lines in a paragraph.

Flags: Read / Write

Allowed values: >= 0

Default value: 0

The “populate-all” property

“populate-all” gboolean

If :populate-all is TRUE, the “populate-popup” signal is also emitted for touch popups.

Flags: Read / Write

Default value: FALSE

Since: 3.8

The “right-margin” property

“right-margin” gint

The default right margin for text in the text view. Tags in the buffer may override the default.

Note that this property is confusingly named. In CSS terms, the value set here is padding, and it is applied in addition to the padding from the theme.

Don't confuse this property with “margin-right”.

Flags: Read / Write

Allowed values: >= 0

Default value: 0

The “tabs” property

“tabs” PangoTabArray *

Custom tabs for this text.

Flags: Read / Write

The “top-margin” property

“top-margin” gint

The top margin for text in the text view.

Note that this property is confusingly named. In CSS terms, the value set here is padding, and it is applied in addition to the padding from the theme.

Don't confuse this property with “margin-top”.

Flags: Read / Write

Allowed values: >= 0

Default value: 0

Since: 3.18

The “wrap-mode” property

```
  “wrap-mode”                GtkWrapMode
```

Whether to wrap lines never, at word boundaries, or at character boundaries.

Flags: Read / Write

Default value: GTK_WRAP_NONE

Style Property Details

The “error-underline-color” style property

```
  “error-underline-color”    GdkColor *
```

Color with which to draw error-indication underlines.

Flags: Read

Signal Details

The “backspace” signal

```
void
user_function (GtkTextView *text_view,
               gpointer     user_data)
```

The ::backspace signal is a keybinding signal which gets emitted when the user asks for it.

The default bindings for this signal are Backspace and Shift-Backspace.

Parameters

text_view	the object which received the signal
user_data	user data set when the signal handler was connected.

Flags: Action

The “copy-clipboard” signal

```
void
user_function (GtkTextView *text_view,
```

```
                gpointer     user_data)
```

The ::copy-clipboard signal is a keybinding signal which gets emitted to copy the selection to the clipboard.

The default bindings for this signal are Ctrl-c and Ctrl-Insert.

Parameters

text_view	the object which received the signal
user_data	user data set when the signal handler was connected.

Flags: Action

The `"cut-clipboard"` signal

```
void
user_function (GtkTextView *text_view,
               gpointer     user_data)
```

The ::cut-clipboard signal is a keybinding signal which gets emitted to cut the selection to the clipboard.

The default bindings for this signal are Ctrl-x and Shift-Delete.

Parameters

text_view	the object which received the signal
user_data	user data set when the signal handler was connected.

Flags: Action

The `"delete-from-cursor"` signal

```
void
user_function (GtkTextView  *text_view,
               GtkDeleteType type,
               gint          count,
               gpointer      user_data)
```

The ::delete-from-cursor signal is a keybinding signal which gets emitted when the user initiates a text deletion.

If the `type` is `GTK_DELETE_CHARS`, GTK+ deletes the selection if there is one, otherwise it deletes the requested number of characters.

The default bindings for this signal are Delete for deleting a character, Ctrl-Delete for deleting a word and Ctrl-Backspace for deleting a word backwords.

Parameters

text_view	the object which received the signal
type	the granularity of the deletion, as a GtkDeleteType

count	the number of `type` units to delete
user_data	user data set when the signal handler was connected.

Flags: Action

The `"extend-selection"` signal

```
gboolean
user_function (GtkTextView           *text_view,
               GtkTextExtendSelection granularity,
               GtkTextIter           *location,
               GtkTextIter           *start,
               GtkTextIter           *end,
               gpointer               user_data)
```

The ::extend-selection signal is emitted when the selection needs to be extended at `location` .

Parameters

text_view	the object which received the signal
granularity	the granularity type
location	the location where to extend the selection
start	where the selection should start
end	where the selection should end
user_data	user data set when the signal handler was connected.

Returns

`GDK_EVENT_STOP` to stop other handlers from being invoked for the event. `GDK_EVENT_PROPAGATE` to propagate the event further.

Flags: Run Last

Since: 3.16

The `"insert-at-cursor"` signal

```
void
user_function (GtkTextView *text_view,
               gchar       *string,
               gpointer     user_data)
```

The ::insert-at-cursor signal is a keybinding signal which gets emitted when the user initiates the insertion of a fixed string at the cursor.

This signal has no default bindings.

Parameters

text_view	the object which received the signal
string	the string to insert
user_data	user data set when the signal handler was connected.

Flags: Action

The `"insert-emoji"` signal

```
void
user_function (GtkTextView *text_view,
               gpointer     user_data)
```

The ::insert-emoji signal is a keybinding signal which gets emitted to present the Emoji chooser for the `text_view`.

The default bindings for this signal are Ctrl-. and Ctrl-;

Parameters

text_view	the object which received the signal
user_data	user data set when the signal handler was connected.

Flags: Action

Since: 3.22.27

The `"move-cursor"` signal

```
void
user_function (GtkTextView    *text_view,
               GtkMovementStep step,
               gint            count,
               gboolean        extend_selection,
               gpointer        user_data)
```

The ::move-cursor signal is a keybinding signal which gets emitted when the user initiates a cursor movement. If the cursor is not visible in `text_view`, this signal causes the viewport to be moved instead.

Applications should not connect to it, but may emit it with `g_signal_emit_by_name()` if they need to control the cursor programmatically.

The default bindings for this signal come in two variants, the variant with the Shift modifier extends the selection, the variant without the Shift modifer does not. There are too many key combinations to list them all here.

- Arrow keys move by individual characters/lines
- Ctrl-arrow key combinations move by words/paragraphs
- Home/End keys move to the ends of the buffer
- PageUp/PageDown keys move vertically by pages

- Ctrl-PageUp/PageDown keys move horizontally by pages

Parameters

text_view	the object which received the signal
step	the granularity of the move, as a GtkMovementStep
count	the number of `step` units to move
extend_selection	`TRUE` if the move should extend the selection
user_data	user data set when the signal handler was connected.

Flags: Action

The `"move-viewport"` signal

```
void
user_function (GtkTextView  *text_view,
               GtkScrollStep step,
               gint          count,
               gpointer      user_data)
```

The ::move-viewport signal is a keybinding signal which can be bound to key combinations to allow the user to move the viewport, i.e. change what part of the text view is visible in a containing scrolled window.

There are no default bindings for this signal.

Parameters

text_view	the object which received the signal
step	the granularity of the movement, as a GtkScrollStep
count	the number of `step` units to move
user_data	user data set when the signal handler was connected.

Flags: Action

The `"paste-clipboard"` signal

```
void
user_function (GtkTextView *text_view,
               gpointer     user_data)
```

The ::paste-clipboard signal is a keybinding signal which gets emitted to paste the contents of the clipboard into the text view.

The default bindings for this signal are Ctrl-v and Shift-Insert.

Parameters

text_view	the object which received the signal
user_data	user data set when the signal handler was connected.

Flags: Action

The `"populate-popup"` signal

```
void
user_function (GtkTextView *text_view,
               GtkWidget   *popup,
               gpointer     user_data)
```

The ::populate-popup signal gets emitted before showing the context menu of the text view.

If you need to add items to the context menu, connect to this signal and append your items to the `popup` , which will be a GtkMenu in this case.

If "populate-all" is `TRUE`, this signal will also be emitted to populate touch popups. In this case, `popup` will be a different container, e.g. a GtkToolbar.

The signal handler should not make assumptions about the type of `widget` , but check whether `popup` is a GtkMenu or GtkToolbar or another kind of container.

Parameters

text_view	The text view on which the signal is emitted
popup	the container that is being populated
user_data	user data set when the signal handler was connected.

Flags: Run Last

The `"preedit-changed"` signal

```
void
user_function (GtkTextView *text_view,
               gchar       *preedit,
               gpointer     user_data)
```

If an input method is used, the typed text will not immediately be committed to the buffer. So if you are interested in the text, connect to this signal.

This signal is only emitted if the text at the given position is actually editable.

Parameters

text_view	the object which received the signal
preedit	the current preedit string
user_data	user data set when the signal

	handler was connected.

Flags: Action

Since: 2.20

The “select-all” signal

```
void
user_function (GtkTextView *text_view,
               gboolean     select,
               gpointer     user_data)
```

The ::select-all signal is a keybinding signal which gets emitted to select or unselect the complete contents of the text view.

The default bindings for this signal are Ctrl-a and Ctrl-/ for selecting and Shift-Ctrl-a and Ctrl-\ for unselecting.

Parameters

text_view	the object which received the signal
select	TRUE to select, FALSE to unselect
user_data	user data set when the signal handler was connected.

Flags: Action

The “set-anchor” signal

```
void
user_function (GtkTextView *text_view,
               gpointer     user_data)
```

The ::set-anchor signal is a keybinding signal which gets emitted when the user initiates setting the "anchor" mark. The "anchor" mark gets placed at the same position as the "insert" mark.

This signal has no default bindings.

Parameters

text_view	the object which received the signal
user_data	user data set when the signal handler was connected.

Flags: Action

The “toggle-cursor-visible” signal

```
void
user_function (GtkTextView *text_view,
               gpointer     user_data)
```

The ::toggle-cursor-visible signal is a keybinding signal which gets emitted to toggle the “cursor-visible” property.

The default binding for this signal is F7.

Parameters

text_view	the object which received the signal
user_data	user data set when the signal handler was connected.

Flags: Action

The “toggle-overwrite” signal

```
void
user_function (GtkTextView *text_view,
               gpointer     user_data)
```

The ::toggle-overwrite signal is a keybinding signal which gets emitted to toggle the overwrite mode of the text view.

The default bindings for this signal is Insert.

Parameters

text_view	the object which received the signal
user_data	user data set when the signal handler was connected.

Flags: Action

See Also

GtkTextBuffer, GtkTextIter

Tree, List and Icon Grid Widgets

Tree and List Widget Overview — Overview of GtkTreeModel, GtkTreeView, and friends

GtkTreeModel — The tree interface used by GtkTreeView

GtkTreeSelection — The selection object for GtkTreeView

GtkTreeViewColumn — A visible column in a GtkTreeView widget

GtkTreeView — A widget for displaying both trees and lists

GtkTreeView drag-and-drop — Interfaces for drag-and-drop support in GtkTreeView

GtkCellView — A widget displaying a single row of a GtkTreeModel

GtkIconView — A widget which displays a list of icons in a grid

GtkTreeSortable — The interface for sortable models used by GtkTreeView

GtkTreeModelSort — A GtkTreeModel which makes an underlying tree model sortable

GtkTreeModelFilter — A GtkTreeModel which hides parts of an underlying tree model

GtkCellLayout — An interface for packing cells

GtkCellArea — An abstract class for laying out GtkCellRenderers

GtkCellAreaBox — A cell area that renders GtkCellRenderers into a row or a column

GtkCellAreaContext — Stores geometrical information for a series of rows in a GtkCellArea

GtkCellRenderer — An object for rendering a single cell

GtkCellEditable — Interface for widgets that can be used for editing cells

GtkCellRendererAccel — Renders a keyboard accelerator in a cell

GtkCellRendererCombo — Renders a combobox in a cell

GtkCellRendererPixbuf — Renders a pixbuf in a cell

GtkCellRendererProgress — Renders numbers as progress bars

GtkCellRendererSpin — Renders a spin button in a cell

GtkCellRendererText — Renders text in a cell

GtkCellRendererToggle — Renders a toggle button in a cell

GtkCellRendererSpinner — Renders a spinning animation in a cell

GtkListStore — A list-like data structure that can be used with the GtkTreeView

GtkTreeStore — A tree-like data structure that can be used with the GtkTreeView

---

Tree and List Widget Overview

Tree and List Widget Overview — Overview of GtkTreeModel, GtkTreeView, and friends

Overview

To create a tree or list in GTK+, use the GtkTreeModel interface in conjunction with the GtkTreeView widget. This widget is designed around a *Model/View/Controller* design and consists of four major parts:

The tree view widget (GtkTreeView)
The view column (GtkTreeViewColumn)
The cell renderers (GtkCellRenderer etc.)
The model interface (GtkTreeModel)

The *View* is composed of the first three objects, while the last is the *Model*. One of the prime benefits of the MVC design is that multiple views can be created of a single model. For example, a model mapping the file system could be created for a file manager. Many views could be created to display various parts of the file system, but only one copy need be kept in memory.

The purpose of the cell renderers is to provide extensibility to the widget and to allow multiple ways of rendering the same type of data. For example, consider how to render a boolean variable. Should it render it as a string of "True" or "False", "On" or "Off", or should it be rendered as a checkbox?

Creating a model

GTK+ provides two simple models that can be used: the GtkListStore and the GtkTreeStore. GtkListStore is used to model list widgets, while the GtkTreeStore models trees. It is possible to develop a new type of model, but the existing models should be satisfactory for all but the most specialized of situations. Creating the model is quite simple:

```
1                                                   GtkListStore *store = gtk_list_store_new (2,
                                                    G_TYPE_STRING, G_TYPE_BOOLEAN);
```

This creates a list store with two columns: a string column and a boolean column. Typically the 2 is never passed directly like that; usually an enum is created wherein the different columns are enumerated, followed by a token that represents the total number of columns. The next example will illustrate this, only using a tree store instead of a list store. Creating a tree store operates almost exactly the same.

```
1                                                   enum
2                                                   {
3                                                      TITLE_COLUMN,
4                                                      AUTHOR_COLUMN,
5                                                      CHECKED_COLUMN,
6                                                      N_COLUMNS
7                                                   };
8
9                                                   GtkTreeStore *store = gtk_tree_store_new
10                                                  (N_COLUMNS,       /* Total number of columns
11                                                  */
12
                                                    G_TYPE_STRING,   /* Book title
                                                    */

                                                    G_TYPE_STRING,   /* Author
                                                    */

                                                    G_TYPE_BOOLEAN); /* Is checked out?
                                                    */
```

Adding data to the model is done using `gtk_tree_store_set()` or `gtk_list_store_set()`, depending upon which sort of model was created. To do this, a GtkTreeIter must be acquired. The iterator points to the location where data will be added.

Once an iterator has been acquired, `gtk_tree_store_set()` is used to apply data to the part of the model that the iterator points to. Consider the following example:

```
1                                                   GtkTreeIter   iter;
2
3                                                   gtk_tree_store_append (store, &iter,
4                                                   NULL);  /* Acquire an iterator */
5
6                                                   gtk_tree_store_set (store, &iter,
7                                                                       TITLE_COLUMN, "The
8                                                   Principle of Reason",
9                                                                       AUTHOR_COLUMN, "Martin
                                                    Heidegger",
                                                                       CHECKED_COLUMN, FALSE,
                                                                       -1);
```

Notice that the last argument is -1. This is always done because this is a variable-argument function and it needs to know when to stop processing arguments. It can be used to set the data in any or all columns in a given row.

The third argument to `gtk_tree_store_append()` is the parent iterator. It is used to add a row to a GtkTreeStore as a child of an existing row. This means that the new row will only be visible when its parent is visible and in its expanded state. Consider the following example:

```
GtkTreeIter iter1;  /* Parent iter */
GtkTreeIter iter2;  /* Child iter  */

gtk_tree_store_append (store, &iter1, NULL);
/* Acquire a top-level iterator */
gtk_tree_store_set (store, &iter1,
                    TITLE_COLUMN, "The Art of
Computer Programming",
                    AUTHOR_COLUMN, "Donald E.
Knuth",
                    CHECKED_COLUMN, FALSE,
                    -1);

gtk_tree_store_append (store, &iter2,
&iter1);  /* Acquire a child iterator */
gtk_tree_store_set (store, &iter2,
                    TITLE_COLUMN, "Volume 1:
Fundamental Algorithms",
                    -1);

gtk_tree_store_append (store, &iter2,
&iter1);
gtk_tree_store_set (store, &iter2,
                    TITLE_COLUMN, "Volume 2:
Seminumerical Algorithms",
                    -1);

gtk_tree_store_append (store, &iter2,
&iter1);
gtk_tree_store_set (store, &iter2,
                    TITLE_COLUMN, "Volume 3:
Sorting and Searching",
                    -1);
```

Creating the view component

While there are several different models to choose from, there is only one view widget to deal with. It works with either the list or the tree store. Setting up a GtkTreeView is not a difficult matter. It needs a GtkTreeModel to know where to retrieve its data from.

```
GtkWidget *tree;

tree = gtk_tree_view_new_with_model
(GTK_TREE_MODEL (store));
```

Columns and cell renderers

Once the GtkTreeView widget has a model, it will need to know how to display the model. It does this with columns and cell renderers.

Cell renderers are used to draw the data in the tree model in a way. There are a number of cell renderers that come with GTK+, including the GtkCellRendererText, GtkCellRendererPixbuf and the GtkCellRendererToggle. It is relatively easy to write a custom renderer.

A GtkTreeViewColumn is the object that GtkTreeView uses to organize the vertical columns in the tree view. It needs to know the name of the column to label for the user, what type of cell renderer to use, and which piece of

data to retrieve from the model for a given row.

```
GtkCellRenderer *renderer;
GtkTreeViewColumn *column;

renderer = gtk_cell_renderer_text_new ();
column =
gtk_tree_view_column_new_with_attributes
("Author",

renderer,

"text", AUTHOR_COLUMN,

NULL);
gtk_tree_view_append_column (GTK_TREE_VIEW
(tree), column);
```

At this point, all the steps in creating a displayable tree have been covered. The model is created, data is stored in it, a tree view is created and columns are added to it.

Selection handling

Most applications will need to not only deal with displaying data, but also receiving input events from users. To do this, simply get a reference to a selection object and connect to the "changed" signal.

```
/* Prototype for selection handler callback
*/
static void tree_selection_changed_cb
(GtkTreeSelection *selection, gpointer data);

/* Setup the selection handler */
GtkTreeSelection *select;

select = gtk_tree_view_get_selection
(GTK_TREE_VIEW (tree));
gtk_tree_selection_set_mode (select,
GTK_SELECTION_SINGLE);
g_signal_connect (G_OBJECT (select),
"changed",
                  G_CALLBACK
(tree_selection_changed_cb),
                  NULL);
```

Then to retrieve data for the row selected:

```
static void
tree_selection_changed_cb (GtkTreeSelection
*selection, gpointer data)
{
        GtkTreeIter iter;
        GtkTreeModel *model;
        gchar *author;

        if (gtk_tree_selection_get_selected
(selection, &model, &iter))
        {
                gtk_tree_model_get (model,
&iter, AUTHOR_COLUMN, &author, -1);

                g_print ("You selected a book
by %s\n", author);
```

```
                g_free (author);
        }
}
```

Simple Example

Here is a simple example of using a GtkTreeView widget in context of the other widgets. It simply creates a simple model and view, and puts them together. Note that the model is never populated with data — that is left as an exercise for the reader. More information can be found on this in the GtkTreeModel section.

```
enum
{
   TITLE_COLUMN,
   AUTHOR_COLUMN,
   CHECKED_COLUMN,
   N_COLUMNS
};

void
setup_tree (void)
{
   GtkTreeStore *store;
   GtkWidget *tree;
   GtkTreeViewColumn *column;
   GtkCellRenderer *renderer;

   /* Create a model.  We are using the store
model for now, though we
    * could use any other GtkTreeModel */
   store = gtk_tree_store_new (N_COLUMNS,
                               G_TYPE_STRING,
                               G_TYPE_STRING,

G_TYPE_BOOLEAN);

   /* custom function to fill the model with
data */
   populate_tree_model (store);

   /* Create a view */
   tree = gtk_tree_view_new_with_model
(GTK_TREE_MODEL (store));

   /* The view now holds a reference.  We can
get rid of our own
    * reference */
   g_object_unref (G_OBJECT (store));

   /* Create a cell render and arbitrarily
make it red for demonstration
    * purposes */
   renderer = gtk_cell_renderer_text_new ();
   g_object_set (G_OBJECT (renderer),
                 "foreground", "red",
                 NULL);

   /* Create a column, associating the "text"
attribute of the
    * cell_renderer to the first column of
the model */
   column =
gtk_tree_view_column_new_with_attributes
```

```
53                                                  ("Author", renderer,
54
55                                                  "text", AUTHOR_COLUMN,
56
57                                                  NULL);
58
59                                                     /* Add the column to the view. */
60                                                     gtk_tree_view_append_column (GTK_TREE_VIEW
61                                                  (tree), column);
62
63                                                     /* Second column.. title of the book. */
64                                                     renderer = gtk_cell_renderer_text_new ();
65                                                     column =
66                                                  gtk_tree_view_column_new_with_attributes
67                                                  ("Title",
68
                                                    renderer,

                                                    "text", TITLE_COLUMN,

                                                    NULL);
                                                       gtk_tree_view_append_column (GTK_TREE_VIEW
                                                    (tree), column);

                                                       /* Last column.. whether a book is checked
                                                    out. */
                                                       renderer = gtk_cell_renderer_toggle_new
                                                    ();
                                                       column =
                                                    gtk_tree_view_column_new_with_attributes
                                                    ("Checked out",

                                                    renderer,

                                                    "active", CHECKED_COLUMN,

                                                    NULL);
                                                       gtk_tree_view_append_column (GTK_TREE_VIEW
                                                    (tree), column);

                                                       /* Now we can manipulate the view just
                                                    like any other GTK widget */
                                                       ...
                                                    }
```

GtkTreeModel

GtkTreeModel — The tree interface used by GtkTreeView

Functions

gboolean	(*GtkTreeModelForeachFunc) ()
GtkTreePath *	gtk_tree_path_new ()
GtkTreePath *	gtk_tree_path_new_from_string ()
GtkTreePath *	gtk_tree_path_new_from_indices ()
GtkTreePath *	gtk_tree_path_new_from_indicesv ()

gchar *	gtk_tree_path_to_string ()
GtkTreePath *	gtk_tree_path_new_first ()
void	gtk_tree_path_append_index ()
void	gtk_tree_path_prepend_index ()
gint	gtk_tree_path_get_depth ()
gint *	gtk_tree_path_get_indices ()
gint *	gtk_tree_path_get_indices_with_depth ()
void	gtk_tree_path_free ()
GtkTreePath *	gtk_tree_path_copy ()
gint	gtk_tree_path_compare ()
void	gtk_tree_path_next ()
gboolean	gtk_tree_path_prev ()
gboolean	gtk_tree_path_up ()
void	gtk_tree_path_down ()
gboolean	gtk_tree_path_is_ancestor ()
gboolean	gtk_tree_path_is_descendant ()
GtkTreeRowReference *	gtk_tree_row_reference_new ()
GtkTreeRowReference *	gtk_tree_row_reference_new_proxy ()
GtkTreeModel *	gtk_tree_row_reference_get_model ()
GtkTreePath *	gtk_tree_row_reference_get_path ()
gboolean	gtk_tree_row_reference_valid ()
void	gtk_tree_row_reference_free ()
GtkTreeRowReference *	gtk_tree_row_reference_copy ()
void	gtk_tree_row_reference_inserted ()
void	gtk_tree_row_reference_deleted ()
void	gtk_tree_row_reference_reordered ()
GtkTreeIter *	gtk_tree_iter_copy ()
void	gtk_tree_iter_free ()
GtkTreeModelFlags	gtk_tree_model_get_flags ()
gint	gtk_tree_model_get_n_columns ()
GType	gtk_tree_model_get_column_type ()
gboolean	gtk_tree_model_get_iter ()
gboolean	gtk_tree_model_get_iter_from_string ()
gboolean	gtk_tree_model_get_iter_first ()
GtkTreePath *	gtk_tree_model_get_path ()
void	gtk_tree_model_get_value ()
gboolean	gtk_tree_model_iter_next ()
gboolean	gtk_tree_model_iter_previous ()
gboolean	gtk_tree_model_iter_children ()
gboolean	gtk_tree_model_iter_has_child ()
gint	gtk_tree_model_iter_n_children ()
gboolean	gtk_tree_model_iter_nth_child ()
gboolean	gtk_tree_model_iter_parent ()
gchar *	gtk_tree_model_get_string_from_iter ()
void	gtk_tree_model_ref_node ()
void	gtk_tree_model_unref_node ()
void	gtk_tree_model_get ()
void	gtk_tree_model_get_valist ()
void	gtk_tree_model_foreach ()
void	gtk_tree_model_row_changed ()
void	gtk_tree_model_row_inserted ()
void	gtk_tree_model_row_has_child_toggled ()

void	gtk_tree_model_row_deleted ()
void	gtk_tree_model_rows_reordered ()
void	gtk_tree_model_rows_reordered_with_length ()

Signals

void	row-changed	Run Last
void	row-deleted	Run First
void	row-has-child-toggled	Run Last
void	row-inserted	Run First
void	rows-reordered	Run First

Types and Values

	GtkTreeModel
struct	GtkTreeIter
	GtkTreePath
	GtkTreeRowReference
struct	GtkTreeModelIface
enum	GtkTreeModelFlags

Object Hierarchy

```
GBoxed
├── GtkTreeIter
└── GtkTreePath
GInterface
└── GtkTreeModel
```

Prerequisites

GtkTreeModel requires GObject.

Known Derived Interfaces

GtkTreeModel is required by GtkTreeSortable.

Known Implementations

GtkTreeModel is implemented by GtkListStore, GtkTreeModelFilter, GtkTreeModelSort and GtkTreeStore.

Includes

```
#include <gtk/gtk.h>
```

Description

The GtkTreeModel interface defines a generic tree interface for use by the GtkTreeView widget. It is an abstract interface, and is designed to be usable with any appropriate data structure. The programmer just has to implement this interface on their own data type for it to be viewable by a GtkTreeView widget.

The model is represented as a hierarchical tree of strongly-typed, columned data. In other words, the model can be seen as a tree where every node has different values depending on which column is being queried. The type of data found in a column is determined by using the GType system (ie. G_TYPE_INT, GTK_TYPE_BUTTON, G_TYPE_POINTER, etc). The types are homogeneous per column across all nodes. It is important to note that this interface only provides a way of examining a model and observing changes. The implementation of each individual model decides how and if changes are made.

In order to make life simpler for programmers who do not need to write their own specialized model, two generic models are provided — the GtkTreeStore and the GtkListStore. To use these, the developer simply pushes data into these models as necessary. These models provide the data structure as well as all appropriate tree interfaces. As a result, implementing drag and drop, sorting, and storing data is trivial. For the vast majority of trees and lists, these two models are sufficient.

Models are accessed on a node/column level of granularity. One can query for the value of a model at a certain node and a certain column on that node. There are two structures used to reference a particular node in a model. They are the GtkTreePath and the GtkTreeIter ("iter" is short for iterator). Most of the interface consists of operations on a GtkTreeIter.

A path is essentially a potential node. It is a location on a model that may or may not actually correspond to a node on a specific model. The GtkTreePath can be converted into either an array of unsigned integers or a string. The string form is a list of numbers separated by a colon. Each number refers to the offset at that level. Thus, the path `0` refers to the root node and the path `2:4` refers to the fifth child of the third node.

By contrast, a GtkTreeIter is a reference to a specific node on a specific model. It is a generic struct with an integer and three generic pointers. These are filled in by the model in a model-specific way. One can convert a path to an iterator by calling `gtk_tree_model_get_iter()`. These iterators are the primary way of accessing a model and are similar to the iterators used by GtkTextBuffer. They are generally statically allocated on the stack and only used for a short time. The model interface defines a set of operations using them for navigating the model.

It is expected that models fill in the iterator with private data. For example, the GtkListStore model, which is internally a simple linked list, stores a list node in one of the pointers. The GtkTreeModelSort stores an array and an offset in two of the pointers. Additionally, there is an integer field. This field is generally filled with a unique stamp per model. This stamp is for catching errors resulting from using invalid iterators with a model.

The lifecycle of an iterator can be a little confusing at first. Iterators are expected to always be valid for as long as the model is unchanged (and doesn't emit a signal). The model is considered to own all outstanding iterators and nothing needs to be done to free them from the user's point of view. Additionally, some models guarantee that an iterator is valid for as long as the node it refers to is valid (most notably the GtkTreeStore and GtkListStore). Although generally uninteresting, as one always has to allow for the case where iterators do not persist beyond a signal, some very important performance enhancements were made in the sort model. As a result, the GTK_TREE_MODEL_ITERS_PERSIST flag was added to indicate this behavior.

To help show some common operation of a model, some examples are provided. The first example shows three ways of getting the iter at the location `3:2:5`. While the first method shown is easier, the second is much more common, as you often get paths from callbacks.

Acquiring a GtkTreeIter

```
// Three ways of getting the iter pointing to
the location
GtkTreePath *path;
GtkTreeIter iter;
GtkTreeIter parent_iter;

// get the iterator from a string
gtk_tree_model_get_iter_from_string (model,
                                     &iter,

"3:2:5");

// get the iterator from a path
path = gtk_tree_path_new_from_string
("3:2:5");
gtk_tree_model_get_iter (model, &iter, path);
gtk_tree_path_free (path);

// walk the tree to find the iterator
gtk_tree_model_iter_nth_child (model, &iter,
                               NULL, 3);
parent_iter = iter;
gtk_tree_model_iter_nth_child (model, &iter,
                               &parent_iter,
2);
parent_iter = iter;
gtk_tree_model_iter_nth_child (model, &iter,
                               &parent_iter,
5);
```

This second example shows a quick way of iterating through a list and getting a string and an integer from each row. The `populate_model()` function used below is not shown, as it is specific to the GtkListStore. For information on how to write such a function, see the GtkListStore documentation.

Reading data from a GtkTreeModel

```
enum
{
  STRING_COLUMN,
  INT_COLUMN,
  N_COLUMNS
};

...

GtkTreeModel *list_store;
GtkTreeIter iter;
gboolean valid;
gint row_count = 0;

// make a new list_store
list_store = gtk_list_store_new (N_COLUMNS,

G_TYPE_STRING,
                                 G_TYPE_INT);

// Fill the list store with data
populate_model (list_store);

// Get the first iter in the list, check it
```

```
25                                                  is valid and walk
26                                                  // through the list, reading each row.
27
28                                                  valid = gtk_tree_model_get_iter_first
29                                                  (list_store,
30
31                                                  &iter);
32                                                  while (valid)
33                                                   {
34                                                     gchar *str_data;
35                                                     gint   int_data;
36
37                                                     // Make sure you terminate calls to
38                                                  gtk_tree_model_get() with a “-1” value
39                                                     gtk_tree_model_get (list_store, &iter,
40                                                                         STRING_COLUMN,
41                                                  &str_data,
42                                                                         INT_COLUMN, &int_data,
43                                                                         -1);
44
45                                                     // Do something with the data
46                                                     g_print ("Row %d: (%s,%d)\n",
47                                                              row_count, str_data, int_data);
                                                       g_free (str_data);

                                                       valid = gtk_tree_model_iter_next
                                                  (list_store,
                                                                                        &iter);
                                                       row_count++;
                                                    }
```

The GtkTreeModel interface contains two methods for reference counting: `gtk_tree_model_ref_node()` and `gtk_tree_model_unref_node()`. These two methods are optional to implement. The reference counting is meant as a way for views to let models know when nodes are being displayed. GtkTreeView will take a reference on a node when it is visible, which means the node is either in the toplevel or expanded. Being displayed does not mean that the node is currently directly visible to the user in the viewport. Based on this reference counting scheme a caching model, for example, can decide whether or not to cache a node based on the reference count. A file-system based model would not want to keep the entire file hierarchy in memory, but just the folders that are currently expanded in every current view.

When working with reference counting, the following rules must be taken into account:

- Never take a reference on a node without owning a reference on its parent. This means that all parent nodes of a referenced node must be referenced as well.
- Outstanding references on a deleted node are not released. This is not possible because the node has already been deleted by the time the row-deleted signal is received.
- Models are not obligated to emit a signal on rows of which none of its siblings are referenced. To phrase this differently, signals are only required for levels in which nodes are referenced. For the root level however, signals must be emitted at all times (however the root level is always referenced when any view is attached).

Functions

GtkTreeModelForeachFunc ()

`gboolean`

```
(*GtkTreeModelForeachFunc) (GtkTreeModel *model,
                            GtkTreePath *path,
                            GtkTreeIter *iter,
                            gpointer data);
```

Type of the callback passed to `gtk_tree_model_foreach()` to iterate over the rows in a tree model.

Parameters

model	the GtkTreeModel being iterated	
path	the current GtkTreePath	
iter	the current GtkTreeIter	
data	The user data passed to `gtk_tree_model_foreach()`.	[closure]

Returns

`TRUE` to stop iterating, `FALSE` to continue

---

gtk_tree_path_new ()

```
GtkTreePath *
gtk_tree_path_new (void);
```

Creates a new GtkTreePath. This refers to a row.

Returns

A newly created GtkTreePath.

---

gtk_tree_path_new_from_string ()

```
GtkTreePath *
gtk_tree_path_new_from_string (const gchar *path);
```

Creates a new GtkTreePath initialized to `path` .

`path` is expected to be a colon separated list of numbers. For example, the string “10:4:0” would create a path of depth 3 pointing to the 11th child of the root node, the 5th child of that 11th child, and the 1st child of that 5th child. If an invalid path string is passed in, `NULL` is returned.

Parameters

path	The string representation of a path

Returns

A newly-created GtkTreePath, or `NULL`

---

gtk_tree_path_new_from_indices ()

```
GtkTreePath *
gtk_tree_path_new_from_indices (gint first_index,
                                ...);
```

Creates a new path with `first_index` and `varargs` as indices.

Parameters

first_index	first integer
...	list of integers terminated by -1

Returns

A newly created GtkTreePath

Since: 2.2

---

gtk_tree_path_new_from_indicesv ()

```
GtkTreePath *
gtk_tree_path_new_from_indicesv (gint *indices,
                                 gsize length);
```

Creates a new path with the given `indices` array of `length` .

[rename-to gtk_tree_path_new_from_indices]

Parameters

indices	array of indices.	[array length=length]
length	length of `indices` array	

Returns

A newly created GtkTreePath

Since: 3.12

---

gtk_tree_path_to_string ()

```
gchar *
gtk_tree_path_to_string (GtkTreePath *path);
```

Generates a string representation of the path.

This string is a ":" separated list of numbers. For example, "4:10:0:3" would be an acceptable return value for this string.

Parameters

path	A GtkTreePath

Returns

A newly-allocated string. Must be freed with `g_free()`.

---

gtk_tree_path_new_first ()

```
GtkTreePath *
gtk_tree_path_new_first (void);
```

Creates a new GtkTreePath.

The string representation of this path is “0”.

Returns

A new GtkTreePath

---

gtk_tree_path_append_index ()

```
void
gtk_tree_path_append_index (GtkTreePath *path,
                            gint index_);
```

Appends a new index to a path.

As a result, the depth of the path is increased.

Parameters

path	a GtkTreePath
index_	the index

---

gtk_tree_path_prepend_index ()

```
void
gtk_tree_path_prepend_index (GtkTreePath *path,
                             gint index_);
```

Prepends a new index to a path.

As a result, the depth of the path is increased.

Parameters

path	a GtkTreePath
index_	the index

---

gtk_tree_path_get_depth ()

```
gint
gtk_tree_path_get_depth (GtkTreePath *path);
```

Returns the current depth of `path` .

Parameters

path	a GtkTreePath

Returns

The depth of `path`

---

gtk_tree_path_get_indices ()

```
gint *
gtk_tree_path_get_indices (GtkTreePath *path);
```

Returns the current indices of `path` .

This is an array of integers, each representing a node in a tree. This value should not be freed.

The length of the array can be obtained with `gtk_tree_path_get_depth()`.

[skip]

Parameters

path	a GtkTreePath

Returns

The current indices, or `NULL`

---

gtk_tree_path_get_indices_with_depth ()

```
gint *
gtk_tree_path_get_indices_with_depth (GtkTreePath *path,
                                      gint *depth);
```

Returns the current indices of `path` .

This is an array of integers, each representing a node in a tree. It also returns the number of elements in the array. The array should not be freed.

[rename-to gtk_tree_path_get_indices]

Parameters

path	a GtkTreePath	
depth	return location for number of elements returned in the integer array, or `NULL`.	[out][allow-none]

Returns

The current indices, or `NULL`.

[array length=depth][transfer none]

Since: 3.0

---

gtk_tree_path_free ()

```
void
gtk_tree_path_free (GtkTreePath *path);
```

Frees `path` . If `path` is `NULL`, it simply returns.

Parameters

path	a GtkTreePath.	[allow-none]

---

gtk_tree_path_copy ()

```
GtkTreePath *
gtk_tree_path_copy (const GtkTreePath *path);
```

Creates a new GtkTreePath as a copy of `path` .

Parameters

path	a GtkTreePath

Returns

a new GtkTreePath

---

gtk_tree_path_compare ()

```
gint
gtk_tree_path_compare (const GtkTreePath *a,
                       const GtkTreePath *b);
```

Compares two paths.

If a appears before b in a tree, then -1 is returned. If b appears before a , then 1 is returned. If the two nodes are equal, then 0 is returned.

Parameters

a	a GtkTreePath
b	a GtkTreePath to compare with

Returns

the relative positions of a and b

---

gtk_tree_path_next ()

```
void
gtk_tree_path_next (GtkTreePath *path);
```

Moves the path to point to the next node at the current depth.

Parameters

path	a GtkTreePath

---

gtk_tree_path_prev ()

```
gboolean
gtk_tree_path_prev (GtkTreePath *path);
```

Moves the path to point to the previous node at the current depth, if it exists.

Parameters

path	a GtkTreePath

Returns

TRUE if path has a previous node, and the move was made

---

gtk_tree_path_up ()

```
gboolean
gtk_tree_path_up (GtkTreePath *path);
```

Moves the `path` to point to its parent node, if it has a parent.

Parameters

path	a GtkTreePath

Returns

`TRUE` if `path` has a parent, and the move was made

---

gtk_tree_path_down ()

```
void
gtk_tree_path_down (GtkTreePath *path);
```

Moves `path` to point to the first child of the current path.

Parameters

path	a GtkTreePath

---

gtk_tree_path_is_ancestor ()

```
gboolean
gtk_tree_path_is_ancestor (GtkTreePath *path,
                           GtkTreePath *descendant);
```

Returns `TRUE` if `descendant` is a descendant of `path` .

Parameters

path	a GtkTreePath
descendant	another GtkTreePath

Returns

`TRUE` if `descendant` is contained inside `path`

---

gtk_tree_path_is_descendant ()

```
gboolean
gtk_tree_path_is_descendant (GtkTreePath *path,
```

```
                                GtkTreePath *ancestor);
```

Returns `TRUE` if `path` is a descendant of `ancestor` .

Parameters

path	a GtkTreePath
ancestor	another GtkTreePath

Returns

`TRUE` if `ancestor` contains `path` somewhere below it

---

gtk_tree_row_reference_new ()

```
GtkTreeRowReference *
gtk_tree_row_reference_new (GtkTreeModel *model,
                            GtkTreePath *path);
```

Creates a row reference based on `path` .

This reference will keep pointing to the node pointed to by `path` , so long as it exists. Any changes that occur on `model` are propagated, and the path is updated appropriately. If `path` isn't a valid path in `model` , then `NULL` is returned.

Parameters

model	a GtkTreeModel
path	a valid GtkTreePath to monitor

Returns

a newly allocated GtkTreeRowReference, or `NULL`

---

gtk_tree_row_reference_new_proxy ()

```
GtkTreeRowReference *
gtk_tree_row_reference_new_proxy (GObject *proxy,
                                  GtkTreeModel *model,
                                  GtkTreePath *path);
```

You do not need to use this function.

Creates a row reference based on `path` .

This reference will keep pointing to the node pointed to by `path` , so long as it exists. If `path` isn't a valid path in `model` , then `NULL` is returned. However, unlike references created with `gtk_tree_row_reference_new()`, it does not listen to the model for changes. The creator of the row reference must do this explicitly using `gtk_tree_row_reference_inserted()`, `gtk_tree_row_reference_deleted()`, `gtk_tree_row_reference_reordered()`.

These functions must be called exactly once per proxy when the corresponding signal on the model is emitted. This single call updates all row references for that proxy. Since built-in GTK+ objects like GtkTreeView already use this mechanism internally, using them as the proxy object will produce unpredictable results. Further more, passing the same object as `model` and `proxy` doesn't work for reasons of internal implementation.

This type of row reference is primarily meant by structures that need to carefully monitor exactly when a row reference updates itself, and is not generally needed by most applications.

Parameters

proxy	a proxy GObject
model	a GtkTreeModel
path	a valid GtkTreePath to monitor

Returns

a newly allocated GtkTreeRowReference, or `NULL`

---

gtk_tree_row_reference_get_model ()

```
GtkTreeModel *
gtk_tree_row_reference_get_model (GtkTreeRowReference *reference);
```

Returns the model that the row reference is monitoring.

Parameters

reference	a GtkTreeRowReference

Returns

the model.

[transfer none]

Since: 2.8

---

gtk_tree_row_reference_get_path ()

```
GtkTreePath *
gtk_tree_row_reference_get_path (GtkTreeRowReference *reference);
```

Returns a path that the row reference currently points to, or `NULL` if the path pointed to is no longer valid.

Parameters

reference	a GtkTreeRowReference

Returns

a current path, or NULL.

[nullable][transfer full]

---

gtk_tree_row_reference_valid ()

```
gboolean
gtk_tree_row_reference_valid (GtkTreeRowReference *reference);
```

Returns TRUE if the reference is non-NULL and refers to a current valid path.

Parameters

reference	a GtkTreeRowReference, or NULL.	[allow-none]

Returns

TRUE if reference points to a valid path

---

gtk_tree_row_reference_free ()

```
void
gtk_tree_row_reference_free (GtkTreeRowReference *reference);
```

Free's reference . reference may be NULL

Parameters

reference	a GtkTreeRowReference, or NULL.	[allow-none]

---

gtk_tree_row_reference_copy ()

```
GtkTreeRowReference *
gtk_tree_row_reference_copy (GtkTreeRowReference *reference);
```

Copies a GtkTreeRowReference.

Parameters

reference	a GtkTreeRowReference

Returns

a copy of reference

Since: 2.2

gtk_tree_row_reference_inserted ()

```
void
gtk_tree_row_reference_inserted (GObject *proxy,
                                 GtkTreePath *path);
```

Lets a set of row reference created by `gtk_tree_row_reference_new_proxy()` know that the model emitted the "row-inserted" signal.

Parameters

proxy	a GObject
path	the row position that was inserted

gtk_tree_row_reference_deleted ()

```
void
gtk_tree_row_reference_deleted (GObject *proxy,
                                GtkTreePath *path);
```

Lets a set of row reference created by `gtk_tree_row_reference_new_proxy()` know that the model emitted the "row-deleted" signal.

Parameters

proxy	a GObject
path	the path position that was deleted

gtk_tree_row_reference_reordered ()

```
void
gtk_tree_row_reference_reordered (GObject *proxy,
                                  GtkTreePath *path,
                                  GtkTreeIter *iter,
                                  gint *new_order);
```

Lets a set of row reference created by `gtk_tree_row_reference_new_proxy()` know that the model emitted the "rows-reordered" signal.

[skip]

Parameters

proxy	a GObject
path	the parent path of the reordered signal
iter	the iter pointing to the parent of the

new_order	reordered the new order of rows.	[array]

gtk_tree_iter_copy ()

```
GtkTreeIter *
gtk_tree_iter_copy (GtkTreeIter *iter);
```

Creates a dynamically allocated tree iterator as a copy of `iter` .

This function is not intended for use in applications, because you can just copy the structs by value (`GtkTreeIter new_iter = iter;`). You must free this iter with `gtk_tree_iter_free()`.

Parameters

iter	a GtkTreeIter

Returns

a newly-allocated copy of `iter`

gtk_tree_iter_free ()

```
void
gtk_tree_iter_free (GtkTreeIter *iter);
```

Frees an iterator that has been allocated by `gtk_tree_iter_copy()`.

This function is mainly used for language bindings.

Parameters

iter	a dynamically allocated tree iterator

gtk_tree_model_get_flags ()

```
GtkTreeModelFlags
gtk_tree_model_get_flags (GtkTreeModel *tree_model);
```

Returns a set of flags supported by this interface.

The flags are a bitwise combination of GtkTreeModelFlags. The flags supported should not change during the lifetime of the `tree_model` .

Parameters

tree_model	a GtkTreeModel

Returns

the flags supported by this interface

---

gtk_tree_model_get_n_columns ()

```
gint
gtk_tree_model_get_n_columns (GtkTreeModel *tree_model);
```

Returns the number of columns supported by `tree_model`.

Parameters

tree_model	a GtkTreeModel

Returns

the number of columns

---

gtk_tree_model_get_column_type ()

```
GType
gtk_tree_model_get_column_type (GtkTreeModel *tree_model,
                                gint index_);
```

Returns the type of the column.

Parameters

tree_model	a GtkTreeModel
index_	the column index

Returns

the type of the column

---

gtk_tree_model_get_iter ()

```
gboolean
gtk_tree_model_get_iter (GtkTreeModel *tree_model,
                         GtkTreeIter *iter,
                         GtkTreePath *path);
```

Sets `iter` to a valid iterator pointing to `path`. If `path` does not exist, `iter` is set to an invalid iterator and `FALSE` is returned.

Parameters

tree_model	a GtkTreeModel	
iter	the uninitialized GtkTreeIter.	[out]
path	the GtkTreePath	

Returns

TRUE, if iter was set

---

gtk_tree_model_get_iter_from_string ()

```
gboolean
gtk_tree_model_get_iter_from_string (GtkTreeModel *tree_model,
                                     GtkTreeIter *iter,
                                     const gchar *path_string);
```

Sets iter to a valid iterator pointing to path_string , if it exists. Otherwise, iter is left invalid and FALSE is returned.

Parameters

tree_model	a GtkTreeModel	
iter	an uninitialized GtkTreeIter.	[out]
path_string	a string representation of a GtkTreePath	

Returns

TRUE, if iter was set

---

gtk_tree_model_get_iter_first ()

```
gboolean
gtk_tree_model_get_iter_first (GtkTreeModel *tree_model,
                               GtkTreeIter *iter);
```

Initializes iter with the first iterator in the tree (the one at the path "0") and returns TRUE. Returns FALSE if the tree is empty.

Parameters

tree_model	a GtkTreeModel	
iter	the uninitialized GtkTreeIter.	[out]

Returns

TRUE, if iter was set

---

gtk_tree_model_get_path ()

```
GtkTreePath *
gtk_tree_model_get_path (GtkTreeModel *tree_model,
                         GtkTreeIter *iter);
```

Returns a newly-created GtkTreePath referenced by `iter`.

This path should be freed with `gtk_tree_path_free()`.

Parameters

tree_model	a GtkTreeModel	
iter	the GtkTreeIter	

Returns

a newly-created GtkTreePath

---

gtk_tree_model_get_value ()

```
void
gtk_tree_model_get_value (GtkTreeModel *tree_model,
                          GtkTreeIter *iter,
                          gint column,
                          GValue *value);
```

Initializes and sets `value` to that at `column`.

When done with `value`, `g_value_unset()` needs to be called to free any allocated memory.

Parameters

tree_model	a GtkTreeModel	
iter	the GtkTreeIter	
column	the column to lookup the value at	
value	an empty GValue to set.	[out][transfer none]

---

gtk_tree_model_iter_next ()

```
gboolean
gtk_tree_model_iter_next (GtkTreeModel *tree_model,
                          GtkTreeIter *iter);
```

Sets `iter` to point to the node following it at the current level.

If there is no next `iter`, `FALSE` is returned and `iter` is set to be invalid.

Parameters

tree_model	a GtkTreeModel	
iter	the GtkTreeIter.	[in]

Returns

`TRUE` if `iter` has been changed to the next node

---

gtk_tree_model_iter_previous ()

```
gboolean
gtk_tree_model_iter_previous (GtkTreeModel *tree_model,
                              GtkTreeIter *iter);
```

Sets `iter` to point to the previous node at the current level.

If there is no previous `iter` , `FALSE` is returned and `iter` is set to be invalid.

Parameters

tree_model	a GtkTreeModel	
iter	the GtkTreeIter.	[in]

Returns

`TRUE` if `iter` has been changed to the previous node

Since: 3.0

---

gtk_tree_model_iter_children ()

```
gboolean
gtk_tree_model_iter_children (GtkTreeModel *tree_model,
                              GtkTreeIter *iter,
                              GtkTreeIter *parent);
```

Sets `iter` to point to the first child of `parent` .

If `parent` has no children, `FALSE` is returned and `iter` is set to be invalid. `parent` will remain a valid node after this function has been called.

If `parent` is `NULL` returns the first node, equivalent to `gtk_tree_model_get_iter_first (tree_model, iter);`

Parameters

tree_model	a GtkTreeModel	
iter	the new GtkTreeIter to be set to the child.	[out]

parent	the GtkTreeIter, or NULL.	[allow-none]

Returns

TRUE, if iter has been set to the first child

---

gtk_tree_model_iter_has_child ()

```
gboolean
gtk_tree_model_iter_has_child (GtkTreeModel *tree_model,
                               GtkTreeIter *iter);
```

Returns TRUE if iter has children, FALSE otherwise.

Parameters

tree_model	a GtkTreeModel	
iter	the GtkTreeIter to test for children	

Returns

TRUE if iter has children

---

gtk_tree_model_iter_n_children ()

```
gint
gtk_tree_model_iter_n_children (GtkTreeModel *tree_model,
                                GtkTreeIter *iter);
```

Returns the number of children that iter has.

As a special case, if iter is NULL, then the number of toplevel nodes is returned.

Parameters

tree_model	a GtkTreeModel	
iter	the GtkTreeIter, or NULL.	[allow-none]

Returns

the number of children of iter

---

gtk_tree_model_iter_nth_child ()

```
gboolean
gtk_tree_model_iter_nth_child (GtkTreeModel *tree_model,
```

```
                                GtkTreeIter *iter,
                                GtkTreeIter *parent,
                                gint n);
```

Sets `iter` to be the child of `parent` , using the given index.

The first index is 0. If `n` is too big, or `parent` has no children, `iter` is set to an invalid iterator and `FALSE` is returned. `parent` will remain a valid node after this function has been called. As a special case, if `parent` is `NULL`, then the `n` -th root node is set.

Parameters

tree_model	a GtkTreeModel	
iter	the GtkTreeIter to set to the nth child.	[out]
parent	the GtkTreeIter to get the child from, or `NULL`.	[allow-none]
n	the index of the desired child	

Returns

`TRUE`, if `parent` has an `n` -th child

---

gtk_tree_model_iter_parent ()

```
gboolean
gtk_tree_model_iter_parent (GtkTreeModel *tree_model,
                            GtkTreeIter *iter,
                            GtkTreeIter *child);
```

Sets `iter` to be the parent of `child` .

If `child` is at the toplevel, and doesn't have a parent, then `iter` is set to an invalid iterator and `FALSE` is returned. `child` will remain a valid node after this function has been called.

`iter` will be initialized before the lookup is performed, so `child` and `iter` cannot point to the same memory location.

Parameters

tree_model	a GtkTreeModel	
iter	the new GtkTreeIter to set to the parent.	[out]
child	the GtkTreeIter	

Returns

`TRUE`, if `iter` is set to the parent of `child`

---

gtk_tree_model_get_string_from_iter ()

```
gchar *
gtk_tree_model_get_string_from_iter (GtkTreeModel *tree_model,
                                     GtkTreeIter *iter);
```

Generates a string representation of the iter.

This string is a ":" separated list of numbers. For example, "4:10:0:3" would be an acceptable return value for this string.

Parameters

tree_model	a GtkTreeModel
iter	a GtkTreeIter

Returns

a newly-allocated string. Must be freed with `g_free()`.

Since: 2.2

---

gtk_tree_model_ref_node ()

```
void
gtk_tree_model_ref_node (GtkTreeModel *tree_model,
                         GtkTreeIter *iter);
```

Lets the tree ref the node.

This is an optional method for models to implement. To be more specific, models may ignore this call as it exists primarily for performance reasons.

This function is primarily meant as a way for views to let caching models know when nodes are being displayed (and hence, whether or not to cache that node). Being displayed means a node is in an expanded branch, regardless of whether the node is currently visible in the viewport. For example, a file-system based model would not want to keep the entire file-hierarchy in memory, just the sections that are currently being displayed by every current view.

A model should be expected to be able to get an iter independent of its reffed state.

Parameters

tree_model	a GtkTreeModel
iter	the GtkTreeIter

---

gtk_tree_model_unref_node ()

```
void
gtk_tree_model_unref_node (GtkTreeModel *tree_model,
                           GtkTreeIter *iter);
```

Lets the tree unref the node.

This is an optional method for models to implement. To be more specific, models may ignore this call as it exists primarily for performance reasons. For more information on what this means, see `gtk_tree_model_ref_node()`.

Please note that nodes that are deleted are not unreffed.

Parameters

tree_model	a GtkTreeModel
iter	the GtkTreeIter

---

gtk_tree_model_get ()

```
void
gtk_tree_model_get (GtkTreeModel *tree_model,
                    GtkTreeIter *iter,
                    ...);
```

Gets the value of one or more cells in the row referenced by `iter` . The variable argument list should contain integer column numbers, each column number followed by a place to store the value being retrieved. The list is terminated by a -1. For example, to get a value from column 0 with type `G_TYPE_STRING`, you would write: `gtk_tree_model_get (model, iter, 0, &place_string_here, -1)`, where `place_string_here` is a gchararray to be filled with the string.

Returned values with type `G_TYPE_OBJECT` have to be unreferenced, values with type `G_TYPE_STRING` or `G_TYPE_BOXED` have to be freed. Other values are passed by value.

Parameters

tree_model	a GtkTreeModel
iter	a row in `tree_model`
...	pairs of column number and value return locations, terminated by -1

---

gtk_tree_model_get_valist ()

```
void
gtk_tree_model_get_valist (GtkTreeModel *tree_model,
                           GtkTreeIter *iter,
                           va_list var_args);
```

See `gtk_tree_model_get()`, this version takes a va_list for language bindings to use.

Parameters

tree_model	a GtkTreeModel
iter	a row in `tree_model`

var_args	va_list of column/return location pairs

gtk_tree_model_foreach ()

```
void
gtk_tree_model_foreach (GtkTreeModel *model,
                        GtkTreeModelForeachFunc func,
                        gpointer user_data);
```

Calls func on each node in model in a depth-first fashion.

If `func` returns `TRUE`, then the tree ceases to be walked, and `gtk_tree_model_foreach()` returns.

Parameters

model	a GtkTreeModel	
func	a function to be called on each row.	[scope call]
user_data	user data to passed to `func`	

gtk_tree_model_row_changed ()

```
void
gtk_tree_model_row_changed (GtkTreeModel *tree_model,
                            GtkTreePath *path,
                            GtkTreeIter *iter);
```

Emits the "row-changed" signal on `tree_model`.

Parameters

tree_model	a GtkTreeModel
path	a GtkTreePath pointing to the changed row
iter	a valid GtkTreeIter pointing to the changed row

gtk_tree_model_row_inserted ()

```
void
gtk_tree_model_row_inserted (GtkTreeModel *tree_model,
                             GtkTreePath *path,
                             GtkTreeIter *iter);
```

Emits the "row-inserted" signal on `tree_model`.

Parameters

tree_model	a GtkTreeModel
path	a GtkTreePath pointing to the inserted row
iter	a valid GtkTreeIter pointing to the inserted row

---

gtk_tree_model_row_has_child_toggled ()

```
void
gtk_tree_model_row_has_child_toggled (GtkTreeModel *tree_model,
                                      GtkTreePath *path,
                                      GtkTreeIter *iter);
```

Emits the “row-has-child-toggled” signal on `tree_model`. This should be called by models after the child state of a node changes.

Parameters

tree_model	a GtkTreeModel
path	a GtkTreePath pointing to the changed row
iter	a valid GtkTreeIter pointing to the changed row

---

gtk_tree_model_row_deleted ()

```
void
gtk_tree_model_row_deleted (GtkTreeModel *tree_model,
                            GtkTreePath *path);
```

Emits the “row-deleted” signal on `tree_model`.

This should be called by models after a row has been removed. The location pointed to by `path` should be the location that the row previously was at. It may not be a valid location anymore.

Nodes that are deleted are not unreffed, this means that any outstanding references on the deleted node should not be released.

Parameters

tree_model	a GtkTreeModel
path	a GtkTreePath pointing to the previous location of the deleted row

---

gtk_tree_model_rows_reordered ()

```
void
gtk_tree_model_rows_reordered (GtkTreeModel *tree_model,
                               GtkTreePath *path,
                               GtkTreeIter *iter,
                               gint *new_order);
```

Emits the "rows-reordered" signal on `tree_model`.

This should be called by models when their rows have been reordered.

[skip]

Parameters

tree_model	a GtkTreeModel	
path	a GtkTreePath pointing to the tree node whose children have been reordered	
iter	a valid GtkTreeIter pointing to the node whose children have been reordered, or `NULL` if the depth of `path` is 0	
new_order	an array of integers mapping the current position of each child to its old position before the re-ordering, i.e. `new_order [newpos] = oldpos`	

---

gtk_tree_model_rows_reordered_with_length ()

```
void
gtk_tree_model_rows_reordered_with_length
                               (GtkTreeModel *tree_model,
                                GtkTreePath *path,
                                GtkTreeIter *iter,
                                gint *new_order,
                                gint length);
```

Emits the "rows-reordered" signal on `tree_model`.

This should be called by models when their rows have been reordered.

[rename-to gtk_tree_model_rows_reordered]

Parameters

tree_model	a GtkTreeModel	
path	a GtkTreePath pointing to the tree node whose children have been reordered	
iter	a valid GtkTreeIter pointing to the node whose children have been	[allow-none]

	reordered, or `NULL` if the depth of `path` is 0.	
new_order	an array of integers mapping the current position of each child to its old position before the re-ordering, i.e. `new_order [newpos] = oldpos`.	[array length=length]
length	length of `new_order` array	

Since: 3.10

Types and Values

GtkTreeModel

```
typedef struct _GtkTreeModel GtkTreeModel;
```

---

struct GtkTreeIter

```
struct GtkTreeIter {
  gint stamp;
  gpointer user_data;
  gpointer user_data2;
  gpointer user_data3;
};
```

The GtkTreeIter is the primary structure for accessing a GtkTreeModel. Models are expected to put a unique integer in the `stamp` member, and put model-specific data in the three `user_data` members.

Members

gint `stamp`;	a unique stamp to catch invalid iterators
gpointer `user_data`;	model-specific data
gpointer `user_data2`;	model-specific data
gpointer `user_data3`;	model-specific data

GtkTreePath

```
typedef struct _GtkTreePath GtkTreePath;
```

GtkTreeRowReference

```
typedef struct _GtkTreeRowReference GtkTreeRowReference;
```

A GtkTreeRowReference tracks model changes so that it always refers to the same row (a GtkTreePath refers to

a position, not a fixed row). Create a new GtkTreeRowReference with gtk_tree_row_reference_new().

---

struct GtkTreeModelIface

```
struct GtkTreeModelIface {
  /* Signals */
  void         (* row_changed)           (GtkTreeModel *tree_model,
                                          GtkTreePath  *path,
                                          GtkTreeIter  *iter);
  void         (* row_inserted)          (GtkTreeModel *tree_model,
                                          GtkTreePath  *path,
                                          GtkTreeIter  *iter);
  void         (* row_has_child_toggled) (GtkTreeModel *tree_model,
                                          GtkTreePath  *path,
                                          GtkTreeIter  *iter);
  void         (* row_deleted)           (GtkTreeModel *tree_model,
                                          GtkTreePath  *path);
  void         (* rows_reordered)        (GtkTreeModel *tree_model,
                                          GtkTreePath  *path,
                                          GtkTreeIter  *iter,
                                          gint         *new_order);

  /* Virtual Table */
  GtkTreeModelFlags (* get_flags)  (GtkTreeModel *tree_model);

  gint         (* get_n_columns)   (GtkTreeModel *tree_model);
  GType        (* get_column_type) (GtkTreeModel *tree_model,
                                    gint          index_);
  gboolean     (* get_iter)        (GtkTreeModel *tree_model,
                                    GtkTreeIter  *iter,
                                    GtkTreePath  *path);
  GtkTreePath *(* get_path)        (GtkTreeModel *tree_model,
                                    GtkTreeIter  *iter);
  void         (* get_value)       (GtkTreeModel *tree_model,
                                    GtkTreeIter  *iter,
                                    gint          column,
                                    GValue       *value);
  gboolean     (* iter_next)       (GtkTreeModel *tree_model,
                                    GtkTreeIter  *iter);
  gboolean     (* iter_previous)   (GtkTreeModel *tree_model,
                                    GtkTreeIter  *iter);
  gboolean     (* iter_children)   (GtkTreeModel *tree_model,
                                    GtkTreeIter  *iter,
                                    GtkTreeIter  *parent);
  gboolean     (* iter_has_child)  (GtkTreeModel *tree_model,
                                    GtkTreeIter  *iter);
  gint         (* iter_n_children) (GtkTreeModel *tree_model,
                                    GtkTreeIter  *iter);
  gboolean     (* iter_nth_child)  (GtkTreeModel *tree_model,
                                    GtkTreeIter  *iter,
                                    GtkTreeIter  *parent,
                                    gint          n);
  gboolean     (* iter_parent)     (GtkTreeModel *tree_model,
                                    GtkTreeIter  *iter,
                                    GtkTreeIter  *child);
  void         (* ref_node)        (GtkTreeModel *tree_model,
                                    GtkTreeIter  *iter);
  void         (* unref_node)      (GtkTreeModel *tree_model,
                                    GtkTreeIter  *iter);
};
```

Members

`row_changed` ()	Signal emitted when a row in the model has changed.
`row_inserted` ()	Signal emitted when a new row has been inserted in the model.
`row_has_child_toggled` ()	Signal emitted when a row has gotten the first child row or lost its last child row.
`row_deleted` ()	Signal emitted when a row has been deleted.
`rows_reordered` ()	Signal emitted when the children of a node in the GtkTreeModel have been reordered.
`get_flags` ()	Get GtkTreeModelFlags supported by this interface.
`get_n_columns` ()	Get the number of columns supported by the model.
`get_column_type` ()	Get the type of the column.
`get_iter` ()	Sets iter to a valid iterator pointing to path.
`get_path` ()	Gets a newly-created GtkTreePath referenced by iter.
`get_value` ()	Initializes and sets value to that at column.
`iter_next` ()	Sets iter to point to the node following it at the current level.
`iter_previous` ()	Sets iter to point to the previous node at the current level.
`iter_children` ()	Sets iter to point to the first child of parent.
`iter_has_child` ()	`TRUE` if iter has children, `FALSE` otherwise.
`iter_n_children` ()	Gets the number of children that iter has.
`iter_nth_child` ()	Sets iter to be the child of parent, using the given index.
`iter_parent` ()	Sets iter to be the parent of child.
`ref_node` ()	Lets the tree ref the node.
`unref_node` ()	Lets the tree unref the node.

---

enum GtkTreeModelFlags

These flags indicate various properties of a GtkTreeModel.

They are returned by `gtk_tree_model_get_flags()`, and must be static for the lifetime of the object. A more complete description of GTK_TREE_MODEL_ITERS_PERSIST can be found in the overview of this section.

Members

GTK_TREE_MODEL_ITERS_PERSIST	iterators survive all signals emitted by the tree
GTK_TREE_MODEL_LIST_ONLY	the model is a list only, and never has children

Signal Details

The `"row-changed"` signal

```
void
user_function (GtkTreeModel *tree_model,
               GtkTreePath  *path,
               GtkTreeIter  *iter,
               gpointer      user_data)
```

This signal is emitted when a row in the model has changed.

Parameters

tree_model	the GtkTreeModel on which the signal is emitted
path	a GtkTreePath identifying the changed row
iter	a valid GtkTreeIter pointing to the changed row
user_data	user data set when the signal handler was connected.

Flags: Run Last

The `"row-deleted"` signal

```
void
user_function (GtkTreeModel *tree_model,
               GtkTreePath  *path,
               gpointer      user_data)
```

This signal is emitted when a row has been deleted.

Note that no iterator is passed to the signal handler, since the row is already deleted.

This should be called by models after a row has been removed. The location pointed to by `path` should be the location that the row previously was at. It may not be a valid location anymore.

Parameters

tree_model	the GtkTreeModel on which the signal is emitted
path	a GtkTreePath identifying the row
user_data	user data set when the signal

	handler was connected.

Flags: Run First

The “row-has-child-toggled” signal

```
void
user_function (GtkTreeModel *tree_model,
               GtkTreePath  *path,
               GtkTreeIter  *iter,
               gpointer      user_data)
```

This signal is emitted when a row has gotten the first child row or lost its last child row.

Parameters

tree_model	the GtkTreeModel on which the signal is emitted
path	a GtkTreePath identifying the row
iter	a valid GtkTreeIter pointing to the row
user_data	user data set when the signal handler was connected.

Flags: Run Last

The “row-inserted” signal

```
void
user_function (GtkTreeModel *tree_model,
               GtkTreePath  *path,
               GtkTreeIter  *iter,
               gpointer      user_data)
```

This signal is emitted when a new row has been inserted in the model.

Note that the row may still be empty at this point, since it is a common pattern to first insert an empty row, and then fill it with the desired values.

Parameters

tree_model	the GtkTreeModel on which the signal is emitted
path	a GtkTreePath identifying the new row
iter	a valid GtkTreeIter pointing to the new row
user_data	user data set when the signal handler was connected.

Flags: Run First

The “rows-reordered” signal

```
void
user_function (GtkTreeModel *tree_model,
               GtkTreePath  *path,
               GtkTreeIter  *iter,
               gpointer      new_order,
               gpointer      user_data)
```

This signal is emitted when the children of a node in the GtkTreeModel have been reordered.

Note that this signal is not emitted when rows are reordered by DND, since this is implemented by removing and then reinserting the row.

[skip]

Parameters

tree_model	the GtkTreeModel on which the signal is emitted
path	a GtkTreePath identifying the tree node whose children have been reordered
iter	a valid GtkTreeIter pointing to the node whose children have been reordered, or `NULL` if the depth of `path` is 0
new_order	an array of integers mapping the current position of each child to its old position before the re-ordering, i.e. `new_order [newpos] = oldpos`
user_data	user data set when the signal handler was connected.

Flags: Run First

See Also

GtkTreeView, GtkTreeStore, GtkListStore, GtkTreeView drag-and-drop GtkTreeSortable

---

GtkTreeSelection

GtkTreeSelection — The selection object for GtkTreeView

Functions

gboolean	(*GtkTreeSelectionFunc) ()
void	(*GtkTreeSelectionForeachFunc) ()

void	gtk_tree_selection_set_mode ()
GtkSelectionMode	gtk_tree_selection_get_mode ()
void	gtk_tree_selection_set_select_function ()
GtkTreeSelectionFunc	gtk_tree_selection_get_select_function ()
gpointer	gtk_tree_selection_get_user_data ()
GtkTreeView *	gtk_tree_selection_get_tree_view ()
gboolean	gtk_tree_selection_get_selected ()
void	gtk_tree_selection_selected_foreach ()
GList *	gtk_tree_selection_get_selected_rows ()
gint	gtk_tree_selection_count_selected_rows ()
void	gtk_tree_selection_select_path ()
void	gtk_tree_selection_unselect_path ()
gboolean	gtk_tree_selection_path_is_selected ()
void	gtk_tree_selection_select_iter ()
void	gtk_tree_selection_unselect_iter ()
gboolean	gtk_tree_selection_iter_is_selected ()
void	gtk_tree_selection_select_all ()
void	gtk_tree_selection_unselect_all ()
void	gtk_tree_selection_select_range ()
void	gtk_tree_selection_unselect_range ()

Properties

GtkSelectionMode	mode	Read / Write

Signals

void	changed	Run First

Types and Values

GtkTreeSelection
GtkTreeSelectionClass

Object Hierarchy

```
GObject
└── GtkTreeSelection
```

Includes

```
#include <gtk/gtk.h>
```

Description

The GtkTreeSelection object is a helper object to manage the selection for a GtkTreeView widget. The GtkTreeSelection object is automatically created when a new GtkTreeView widget is created, and cannot exist independently of this widget. The primary reason the GtkTreeSelection objects exists is for cleanliness of code and API. That is, there is no conceptual reason all these functions could not be methods on the GtkTreeView

widget instead of a separate function.

The GtkTreeSelection object is gotten from a GtkTreeView by calling `gtk_tree_view_get_selection()`. It can be manipulated to check the selection status of the tree, as well as select and deselect individual rows. Selection is done completely view side. As a result, multiple views of the same model can have completely different selections. Additionally, you cannot change the selection of a row on the model that is not currently displayed by the view without expanding its parents first.

One of the important things to remember when monitoring the selection of a view is that the "changed" signal is mostly a hint. That is, it may only emit one signal when a range of rows is selected. Additionally, it may on occasion emit a "changed" signal when nothing has happened (mostly as a result of programmers calling select_row on an already selected row).

Functions

GtkTreeSelectionFunc ()

```
gboolean
(*GtkTreeSelectionFunc) (GtkTreeSelection *selection,
                         GtkTreeModel *model,
                         GtkTreePath *path,
                         gboolean path_currently_selected,
                         gpointer data);
```

A function used by `gtk_tree_selection_set_select_function()` to filter whether or not a row may be selected. It is called whenever a row's state might change. A return value of `TRUE` indicates to `selection` that it is okay to change the selection.

Parameters

selection	A GtkTreeSelection	
model	A GtkTreeModel being viewed	
path	The GtkTreePath of the row in question	
path_currently_selected	`TRUE`, if the path is currently selected	
data	user data.	[closure]

Returns

`TRUE`, if the selection state of the row can be toggled

GtkTreeSelectionForeachFunc ()

```
void
(*GtkTreeSelectionForeachFunc) (GtkTreeModel *model,
                                GtkTreePath *path,
                                GtkTreeIter *iter,
                                gpointer data);
```

A function used by `gtk_tree_selection_selected_foreach()` to map all selected rows. It will be called on every selected row in the view.

Parameters

model	The GtkTreeModel being viewed	
path	The GtkTreePath of a selected row	
iter	A GtkTreeIter pointing to a selected row	
data	user data.	[closure]

---

gtk_tree_selection_set_mode ()

```
void
gtk_tree_selection_set_mode (GtkTreeSelection *selection,
                             GtkSelectionMode type);
```

Sets the selection mode of the `selection` . If the previous type was GTK_SELECTION_MULTIPLE, then the anchor is kept selected, if it was previously selected.

Parameters

selection	A GtkTreeSelection.
type	The selection mode

---

gtk_tree_selection_get_mode ()

```
GtkSelectionMode
gtk_tree_selection_get_mode (GtkTreeSelection *selection);
```

Gets the selection mode for `selection` . See `gtk_tree_selection_set_mode()`.

Parameters

selection	a GtkTreeSelection

Returns

the current selection mode

---

gtk_tree_selection_set_select_function ()

```
void
gtk_tree_selection_set_select_function
                             (GtkTreeSelection *selection,
                              GtkTreeSelectionFunc func,
```

```
                                gpointer data,
                                GDestroyNotify destroy);
```

Sets the selection function.

If set, this function is called before any node is selected or unselected, giving some control over which nodes are selected. The select function should return `TRUE` if the state of the node may be toggled, and `FALSE` if the state of the node should be left unchanged.

Parameters

selection	A GtkTreeSelection.	
func	The selection function. May be `NULL`.	[nullable]
data	The selection function's data. May be `NULL`	
destroy	The destroy function for user data. May be `NULL`	

---

gtk_tree_selection_get_select_function ()

```
GtkTreeSelectionFunc
gtk_tree_selection_get_select_function
                                (GtkTreeSelection *selection);
```

Returns the current selection function.

[skip]

Parameters

selection	A GtkTreeSelection.

Returns

The function.

Since: 2.14

---

gtk_tree_selection_get_user_data ()

```
gpointer
gtk_tree_selection_get_user_data (GtkTreeSelection *selection);
```

Returns the user data for the selection function.

[skip]

Parameters

selection	A GtkTreeSelection.

Returns

The user data.

---

gtk_tree_selection_get_tree_view ()

```
GtkTreeView *
gtk_tree_selection_get_tree_view (GtkTreeSelection *selection);
```

Returns the tree view associated with `selection` .

Parameters

selection	A GtkTreeSelection

Returns

A GtkTreeView.

[transfer none]

---

gtk_tree_selection_get_selected ()

```
gboolean
gtk_tree_selection_get_selected (GtkTreeSelection *selection,
                                 GtkTreeModel **model,
                                 GtkTreeIter *iter);
```

Sets `iter` to the currently selected node if `selection` is set to GTK_SELECTION_SINGLE or GTK_SELECTION_BROWSE. `iter` may be NULL if you just want to test if `selection` has any selected nodes. `model` is filled with the current model as a convenience. This function will not work if you use `selection` is GTK_SELECTION_MULTIPLE.

Parameters

selection	A GtkTreeSelection.	
model	A pointer to set to the GtkTreeModel, or NULL.	[out][allow-none][transfer none]
iter	The GtkTreeIter, or NULL.	[out][allow-none]

Returns

TRUE, if there is a selected node.

---

gtk_tree_selection_selected_foreach ()

```
void
gtk_tree_selection_selected_foreach (GtkTreeSelection *selection,
                                     GtkTreeSelectionForeachFunc func,
                                     gpointer data);
```

Calls a function for each selected node. Note that you cannot modify the tree or selection from within this function. As a result, `gtk_tree_selection_get_selected_rows()` might be more useful.

Parameters

selection	A GtkTreeSelection.	
func	The function to call for each selected node.	[scope call]
data	user data to pass to the function.	

---

gtk_tree_selection_get_selected_rows ()

```
GList *
gtk_tree_selection_get_selected_rows (GtkTreeSelection *selection,
                                      GtkTreeModel **model);
```

Creates a list of path of all selected rows. Additionally, if you are planning on modifying the model after calling this function, you may want to convert the returned list into a list of GtkTreeRowReferences. To do this, you can use `gtk_tree_row_reference_new()`.

To free the return value, use:

```
1                                       g_list_free_full (list, (GDestroyNotify)
                                        gtk_tree_path_free);
```

Parameters

selection	A GtkTreeSelection.	
model	A pointer to set to the GtkTreeModel, or `NULL`.	[out][allow-none][transfer none]

Returns

A GList containing a GtkTreePath for each selected row.

[element-type GtkTreePath][transfer full]

Since: 2.2

---

gtk_tree_selection_count_selected_rows ()

```
gint
gtk_tree_selection_count_selected_rows
                                (GtkTreeSelection *selection);
```

Returns the number of rows that have been selected in `tree` .

Parameters

selection	A GtkTreeSelection.

Returns

The number of rows selected.

Since: 2.2

---

gtk_tree_selection_select_path ()

```
void
gtk_tree_selection_select_path (GtkTreeSelection *selection,
                                GtkTreePath *path);
```

Select the row at `path` .

Parameters

selection	A GtkTreeSelection.
path	The GtkTreePath to be selected.

---

gtk_tree_selection_unselect_path ()

```
void
gtk_tree_selection_unselect_path (GtkTreeSelection *selection,
                                  GtkTreePath *path);
```

Unselects the row at `path` .

Parameters

selection	A GtkTreeSelection.
path	The GtkTreePath to be unselected.

---

gtk_tree_selection_path_is_selected ()

```
gboolean
gtk_tree_selection_path_is_selected (GtkTreeSelection *selection,
                                     GtkTreePath *path);
```

Returns `TRUE` if the row pointed to by `path` is currently selected. If `path` does not point to a valid location, `FALSE` is returned

Parameters

selection	A GtkTreeSelection.
path	A GtkTreePath to check selection on.

Returns

`TRUE` if `path` is selected.

---

gtk_tree_selection_select_iter ()

```
void
gtk_tree_selection_select_iter (GtkTreeSelection *selection,
                                GtkTreeIter *iter);
```

Selects the specified iterator.

Parameters

selection	A GtkTreeSelection.
iter	The GtkTreeIter to be selected.

---

gtk_tree_selection_unselect_iter ()

```
void
gtk_tree_selection_unselect_iter (GtkTreeSelection *selection,
                                  GtkTreeIter *iter);
```

Unselects the specified iterator.

Parameters

selection	A GtkTreeSelection.
iter	The GtkTreeIter to be unselected.

---

gtk_tree_selection_iter_is_selected ()

```
gboolean
gtk_tree_selection_iter_is_selected (GtkTreeSelection *selection,
                                     GtkTreeIter *iter);
```

Returns `TRUE` if the row at `iter` is currently selected.

Parameters

selection	A GtkTreeSelection
iter	A valid GtkTreeIter

Returns

`TRUE`, if `iter` is selected

---

gtk_tree_selection_select_all ()

```
void
gtk_tree_selection_select_all (GtkTreeSelection *selection);
```

Selects all the nodes. `selection` must be set to GTK_SELECTION_MULTIPLE mode.

Parameters

selection	A GtkTreeSelection.

---

gtk_tree_selection_unselect_all ()

```
void
gtk_tree_selection_unselect_all (GtkTreeSelection *selection);
```

Unselects all the nodes.

Parameters

selection	A GtkTreeSelection.

---

gtk_tree_selection_select_range ()

```
void
gtk_tree_selection_select_range (GtkTreeSelection *selection,
                                 GtkTreePath *start_path,
                                 GtkTreePath *end_path);
```

Selects a range of nodes, determined by `start_path` and `end_path` inclusive. `selection` must be set to GTK_SELECTION_MULTIPLE mode.

Parameters

selection	A GtkTreeSelection.
start_path	The initial node of the range.
end_path	The final node of the range.

---

gtk_tree_selection_unselect_range ()

```
void
gtk_tree_selection_unselect_range (GtkTreeSelection *selection,
                                   GtkTreePath *start_path,
                                   GtkTreePath *end_path);
```

Unselects a range of nodes, determined by `start_path` and `end_path` inclusive.

Parameters

selection	A GtkTreeSelection.
start_path	The initial node of the range.
end_path	The initial node of the range.

Since: 2.2

Types and Values

GtkTreeSelection

```
typedef struct _GtkTreeSelection GtkTreeSelection;
```

GtkTreeSelectionClass

```
typedef struct {
  GObjectClass parent_class;


  void (* changed) (GtkTreeSelection *selection);
} GtkTreeSelectionClass;
```

Members

`changed` ()	Signal emitted whenever the selection has (possibly) changed.

Property Details

The `"mode"` property

`"mode"` `GtkSelectionMode`

Selection mode. See `gtk_tree_selection_set_mode()` for more information on this property.

Flags: Read / Write

Default value: GTK_SELECTION_SINGLE

Since: 3.2

Signal Details

The "changed" signal

```
void
user_function (GtkTreeSelection *treeselection,
               gpointer          user_data)
```

Emitted whenever the selection has (possibly) changed. Please note that this signal is mostly a hint. It may only be emitted once when a range of rows are selected, and it may occasionally be emitted when nothing has happened.

Parameters

treeselection	the object which received the signal.
user_data	user data set when the signal handler was connected.

Flags: Run First

See Also

GtkTreeView, GtkTreeViewColumn, GtkTreeModel, GtkTreeSortable, GtkTreeModelSort, GtkListStore, GtkTreeStore, GtkCellRenderer, GtkCellEditable, GtkCellRendererPixbuf, GtkCellRendererText, GtkCellRendererToggle, GtkTreeView drag-and-drop

---

GtkTreeViewColumn

GtkTreeViewColumn — A visible column in a GtkTreeView widget

Functions

void	(*GtkTreeCellDataFunc) ()
GtkTreeViewColumn *	gtk_tree_view_column_new ()
GtkTreeViewColumn *	gtk_tree_view_column_new_with_area ()
GtkTreeViewColumn *	gtk_tree_view_column_new_with_attributes ()
void	gtk_tree_view_column_pack_start ()
void	gtk_tree_view_column_pack_end ()
void	gtk_tree_view_column_clear ()
void	gtk_tree_view_column_add_attribute ()
void	gtk_tree_view_column_set_attributes ()
void	gtk_tree_view_column_set_cell_data_func ()
void	gtk_tree_view_column_clear_attributes ()
void	gtk_tree_view_column_set_spacing ()
gint	gtk_tree_view_column_get_spacing ()

void	gtk_tree_view_column_set_visible ()
gboolean	gtk_tree_view_column_get_visible ()
void	gtk_tree_view_column_set_resizable ()
gboolean	gtk_tree_view_column_get_resizable ()
void	gtk_tree_view_column_set_sizing ()
GtkTreeViewColumnSizing	gtk_tree_view_column_get_sizing ()
gint	gtk_tree_view_column_get_width ()
gint	gtk_tree_view_column_get_fixed_width ()
void	gtk_tree_view_column_set_fixed_width ()
void	gtk_tree_view_column_set_min_width ()
gint	gtk_tree_view_column_get_min_width ()
void	gtk_tree_view_column_set_max_width ()
gint	gtk_tree_view_column_get_max_width ()
void	gtk_tree_view_column_clicked ()
void	gtk_tree_view_column_set_title ()
const gchar *	gtk_tree_view_column_get_title ()
void	gtk_tree_view_column_set_expand ()
gboolean	gtk_tree_view_column_get_expand ()
void	gtk_tree_view_column_set_clickable ()
gboolean	gtk_tree_view_column_get_clickable ()
void	gtk_tree_view_column_set_widget ()
GtkWidget *	gtk_tree_view_column_get_widget ()
GtkWidget *	gtk_tree_view_column_get_button ()
void	gtk_tree_view_column_set_alignment ()
gfloat	gtk_tree_view_column_get_alignment ()
void	gtk_tree_view_column_set_reorderable ()
gboolean	gtk_tree_view_column_get_reorderable ()
void	gtk_tree_view_column_set_sort_column_id ()
gint	gtk_tree_view_column_get_sort_column_id ()
void	gtk_tree_view_column_set_sort_indicator ()
gboolean	gtk_tree_view_column_get_sort_indicator ()
void	gtk_tree_view_column_set_sort_order ()
GtkSortType	gtk_tree_view_column_get_sort_order ()
void	gtk_tree_view_column_cell_set_cell_data ()
void	gtk_tree_view_column_cell_get_size ()
gboolean	gtk_tree_view_column_cell_get_position ()
gboolean	gtk_tree_view_column_cell_is_visible ()
void	gtk_tree_view_column_focus_cell ()
void	gtk_tree_view_column_queue_resize ()
GtkWidget *	gtk_tree_view_column_get_tree_view ()
gint	gtk_tree_view_column_get_x_offset ()

Properties

gfloat	alignment	Read / Write
GtkCellArea *	cell-area	Read / Write / Construct Only
gboolean	clickable	Read / Write
gboolean	expand	Read / Write
gint	fixed-width	Read / Write
gint	max-width	Read / Write
gint	min-width	Read / Write

gboolean	reorderable	Read / Write
gboolean	resizable	Read / Write
GtkTreeViewColumnSizing	sizing	Read / Write
gint	sort-column-id	Read / Write
gboolean	sort-indicator	Read / Write
GtkSortType	sort-order	Read / Write
gint	spacing	Read / Write
gchar *	title	Read / Write
gboolean	visible	Read / Write
GtkWidget *	widget	Read / Write
gint	width	Read
gint	x-offset	Read

Signals

void	clicked	Run Last

Types and Values

enum	GtkTreeViewColumnSizing
struct	GtkTreeViewColumn

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkTreeViewColumn
```

Implemented Interfaces

GtkTreeViewColumn implements GtkCellLayout and GtkBuildable.

Includes

```
#include <gtk/gtk.h>
```

Description

The GtkTreeViewColumn object represents a visible column in a GtkTreeView widget. It allows to set properties of the column header, and functions as a holding pen for the cell renderers which determine how the data in the column is displayed.

Please refer to the tree widget conceptual overview for an overview of all the objects and data types related to the tree widget and how they work together.

Functions

GtkTreeCellDataFunc ()

```
void
(*GtkTreeCellDataFunc) (GtkTreeViewColumn *tree_column,
                        GtkCellRenderer *cell,
                        GtkTreeModel *tree_model,
                        GtkTreeIter *iter,
                        gpointer data);
```

A function to set the properties of a cell instead of just using the straight mapping between the cell and the model. This is useful for customizing the cell renderer. For example, a function might get an integer from the `tree_model` , and render it to the “text” attribute of “cell” by converting it to its written equivalent. This is set by calling `gtk_tree_view_column_set_cell_data_func()`

Parameters

tree_column	A GtkTreeViewColumn	
cell	The GtkCellRenderer that is being rendered by `tree_column`	
tree_model	The GtkTreeModel being rendered	
iter	A GtkTreeIter of the current row rendered	
data	user data.	[closure]

---

gtk_tree_view_column_new ()

```
GtkTreeViewColumn *
gtk_tree_view_column_new (void);
```

Creates a new GtkTreeViewColumn.

Returns

A newly created GtkTreeViewColumn.

---

gtk_tree_view_column_new_with_area ()

```
GtkTreeViewColumn *
gtk_tree_view_column_new_with_area (GtkCellArea *area);
```

Creates a new GtkTreeViewColumn using `area` to render its cells.

Parameters

area	the GtkCellArea that the newly created column should use to layout cells.

Returns

A newly created GtkTreeViewColumn.

Since: 3.0

---

gtk_tree_view_column_new_with_attributes ()

```
GtkTreeViewColumn *
gtk_tree_view_column_new_with_attributes
                                (const gchar *title,
                                 GtkCellRenderer *cell,
                                 ...);
```

Creates a new GtkTreeViewColumn with a number of default values. This is equivalent to calling `gtk_tree_view_column_set_title()`, `gtk_tree_view_column_pack_start()`, and `gtk_tree_view_column_set_attributes()` on the newly created GtkTreeViewColumn.

Here's a simple example:

```
enum { TEXT_COLUMN, COLOR_COLUMN,
N_COLUMNS };
// ...
{
  GtkTreeViewColumn *column;
  GtkCellRenderer   *renderer =
gtk_cell_renderer_text_new ();

  column =
gtk_tree_view_column_new_with_attributes
("Title",

renderer,

"text", TEXT_COLUMN,

"foreground", COLOR_COLUMN,

NULL);
}
```

Parameters

title	The title to set the header to
cell	The GtkCellRenderer
...	A `NULL`-terminated list of attributes

Returns

A newly created GtkTreeViewColumn.

---

gtk_tree_view_column_pack_start ()

```
void
gtk_tree_view_column_pack_start (GtkTreeViewColumn *tree_column,
```

```
                                GtkCellRenderer *cell,
                                gboolean expand);
```

Packs the `cell` into the beginning of the column. If `expand` is `FALSE`, then the `cell` is allocated no more space than it needs. Any unused space is divided evenly between cells for which `expand` is `TRUE`.

Parameters

tree_column	A GtkTreeViewColumn.
cell	The GtkCellRenderer.
expand	`TRUE` if `cell` is to be given extra space allocated to `tree_column` .

---

gtk_tree_view_column_pack_end ()

```
void
gtk_tree_view_column_pack_end (GtkTreeViewColumn *tree_column,
                               GtkCellRenderer *cell,
                               gboolean expand);
```

Adds the `cell` to end of the column. If `expand` is `FALSE`, then the `cell` is allocated no more space than it needs. Any unused space is divided evenly between cells for which `expand` is `TRUE`.

Parameters

tree_column	A GtkTreeViewColumn.
cell	The GtkCellRenderer.
expand	`TRUE` if `cell` is to be given extra space allocated to `tree_column` .

---

gtk_tree_view_column_clear ()

```
void
gtk_tree_view_column_clear (GtkTreeViewColumn *tree_column);
```

Unsets all the mappings on all renderers on the `tree_column` .

Parameters

tree_column	A GtkTreeViewColumn

---

gtk_tree_view_column_add_attribute ()

```
void
gtk_tree_view_column_add_attribute (GtkTreeViewColumn *tree_column,
                                    GtkCellRenderer *cell_renderer,
                                    const gchar *attribute,
                                    gint column);
```

Adds an attribute mapping to the list in `tree_column` . The `column` is the column of the model to get a value from, and the `attribute` is the parameter on `cell_renderer` to be set from the value. So for example if column 2 of the model contains strings, you could have the “text” attribute of a GtkCellRendererText get its values from column 2.

Parameters

tree_column	A GtkTreeViewColumn.
cell_renderer	the GtkCellRenderer to set attributes on
attribute	An attribute on the renderer
column	The column position on the model to get the attribute from.

---

gtk_tree_view_column_set_attributes ()

```
void
gtk_tree_view_column_set_attributes (GtkTreeViewColumn *tree_column,
                                     GtkCellRenderer *cell_renderer,
                                     ...);
```

Sets the attributes in the list as the attributes of `tree_column` . The attributes should be in attribute/column order, as in `gtk_tree_view_column_add_attribute()`. All existing attributes are removed, and replaced with the new attributes.

Parameters

tree_column	A GtkTreeViewColumn
cell_renderer	the GtkCellRenderer we’re setting the attributes of
...	A `NULL`-terminated list of attributes

---

gtk_tree_view_column_set_cell_data_func ()

```
void
gtk_tree_view_column_set_cell_data_func
                                (GtkTreeViewColumn *tree_column,
                                 GtkCellRenderer *cell_renderer,
                                 GtkTreeCellDataFunc func,
                                 gpointer func_data,
                                 GDestroyNotify destroy);
```

Sets the GtkTreeCellDataFunc to use for the column. This function is used instead of the standard attributes mapping for setting the column value, and should set the value of `tree_column` 's cell renderer as appropriate. `func` may be `NULL` to remove an older one.

Parameters

tree_column	A GtkTreeViewColumn

cell_renderer	A GtkCellRenderer	
func	The GtkTreeCellDataFunc to use.	[allow-none]
func_data	The user data for `func` .	[closure]
destroy	The destroy notification for `func_data`	

---

gtk_tree_view_column_clear_attributes ()

```
void
gtk_tree_view_column_clear_attributes (GtkTreeViewColumn *tree_column,
                                       GtkCellRenderer *cell_renderer);
```

Clears all existing attributes previously set with `gtk_tree_view_column_set_attributes()`.

Parameters

tree_column	a GtkTreeViewColumn
cell_renderer	a GtkCellRenderer to clear the attribute mapping on.

---

gtk_tree_view_column_set_spacing ()

```
void
gtk_tree_view_column_set_spacing (GtkTreeViewColumn *tree_column,
                                  gint spacing);
```

Sets the spacing field of `tree_column` , which is the number of pixels to place between cell renderers packed into it.

Parameters

tree_column	A GtkTreeViewColumn.
spacing	distance between cell renderers in pixels.

---

gtk_tree_view_column_get_spacing ()

```
gint
gtk_tree_view_column_get_spacing (GtkTreeViewColumn *tree_column);
```

Returns the spacing of `tree_column` .

Parameters

tree_column	A GtkTreeViewColumn.

Returns

the spacing of `tree_column` .

---

gtk_tree_view_column_set_visible ()

```
void
gtk_tree_view_column_set_visible (GtkTreeViewColumn *tree_column,
                                  gboolean visible);
```

Sets the visibility of `tree_column` .

Parameters

tree_column	A GtkTreeViewColumn.
visible	`TRUE` if the `tree_column` is visible.

---

gtk_tree_view_column_get_visible ()

```
gboolean
gtk_tree_view_column_get_visible (GtkTreeViewColumn *tree_column);
```

Returns `TRUE` if `tree_column` is visible.

Parameters

tree_column	A GtkTreeViewColumn.

Returns

whether the column is visible or not. If it is visible, then the tree will show the column.

---

gtk_tree_view_column_set_resizable ()

```
void
gtk_tree_view_column_set_resizable (GtkTreeViewColumn *tree_column,
                                    gboolean resizable);
```

If `resizable` is `TRUE`, then the user can explicitly resize the column by grabbing the outer edge of the column button. If resizable is `TRUE` and sizing mode of the column is GTK_TREE_VIEW_COLUMN_AUTOSIZE, then the sizing mode is changed to GTK_TREE_VIEW_COLUMN_GROW_ONLY.

Parameters

tree_column	A GtkTreeViewColumn
resizable	`TRUE`, if the column can be resized

---

gtk_tree_view_column_get_resizable ()

```
gboolean
gtk_tree_view_column_get_resizable (GtkTreeViewColumn *tree_column);
```

Returns `TRUE` if the `tree_column` can be resized by the end user.

Parameters

tree_column	A GtkTreeViewColumn

Returns

`TRUE`, if the `tree_column` can be resized.

---

gtk_tree_view_column_set_sizing ()

```
void
gtk_tree_view_column_set_sizing (GtkTreeViewColumn *tree_column,
                                 GtkTreeViewColumnSizing type);
```

Sets the growth behavior of `tree_column` to `type` .

Parameters

tree_column	A GtkTreeViewColumn.
type	The GtkTreeViewColumnSizing.

---

gtk_tree_view_column_get_sizing ()

```
GtkTreeViewColumnSizing
gtk_tree_view_column_get_sizing (GtkTreeViewColumn *tree_column);
```

Returns the current type of `tree_column` .

Parameters

tree_column	A GtkTreeViewColumn.

Returns

The type of `tree_column` .

---

gtk_tree_view_column_get_width ()

```
gint
gtk_tree_view_column_get_width (GtkTreeViewColumn *tree_column);
```

Returns the current size of `tree_column` in pixels.

Parameters

tree_column	A GtkTreeViewColumn.

Returns

The current width of `tree_column` .

---

gtk_tree_view_column_get_fixed_width ()

```
gint
gtk_tree_view_column_get_fixed_width (GtkTreeViewColumn *tree_column);
```

Gets the fixed width of the column. This may not be the actual displayed width of the column; for that, use `gtk_tree_view_column_get_width()`.

Parameters

tree_column	A GtkTreeViewColumn.

Returns

The fixed width of the column.

---

gtk_tree_view_column_set_fixed_width ()

```
void
gtk_tree_view_column_set_fixed_width (GtkTreeViewColumn *tree_column,
                                      gint fixed_width);
```

If `fixed_width` is not -1, sets the fixed width of `tree_column` ; otherwise unsets it. The effective value of `fixed_width` is clamped between the minimum and maximum width of the column; however, the value stored in the “fixed-width” property is not clamped. If the column sizing is GTK_TREE_VIEW_COLUMN_GROW_ONLY or GTK_TREE_VIEW_COLUMN_AUTOSIZE, setting a fixed width overrides the automatically calculated width. Note that `fixed_width` is only a hint to GTK+; the width actually allocated to the column may be greater or less than requested.

Along with “expand”, the “fixed-width” property changes when the column is resized by the user.

Parameters

tree_column	A GtkTreeViewColumn.

fixed_width	The new fixed width, in pixels, or -1.

gtk_tree_view_column_set_min_width ()

```
void
gtk_tree_view_column_set_min_width (GtkTreeViewColumn *tree_column,
                                    gint min_width);
```

Sets the minimum width of the `tree_column` . If `min_width` is -1, then the minimum width is unset.

Parameters

tree_column	A GtkTreeViewColumn.
min_width	The minimum width of the column in pixels, or -1.

gtk_tree_view_column_get_min_width ()

```
gint
gtk_tree_view_column_get_min_width (GtkTreeViewColumn *tree_column);
```

Returns the minimum width in pixels of the `tree_column` , or -1 if no minimum width is set.

Parameters

tree_column	A GtkTreeViewColumn.

Returns

The minimum width of the `tree_column` .

gtk_tree_view_column_set_max_width ()

```
void
gtk_tree_view_column_set_max_width (GtkTreeViewColumn *tree_column,
                                    gint max_width);
```

Sets the maximum width of the `tree_column` . If `max_width` is -1, then the maximum width is unset. Note, the column can actually be wider than max width if it’s the last column in a view. In this case, the column expands to fill any extra space.

Parameters

tree_column	A GtkTreeViewColumn.
max_width	The maximum width of the column in pixels, or -1.

gtk_tree_view_column_get_max_width ()

```
gint
gtk_tree_view_column_get_max_width (GtkTreeViewColumn *tree_column);
```

Returns the maximum width in pixels of the `tree_column` , or -1 if no maximum width is set.

Parameters

tree_column	A GtkTreeViewColumn.

Returns

The maximum width of the `tree_column` .

---

gtk_tree_view_column_clicked ()

```
void
gtk_tree_view_column_clicked (GtkTreeViewColumn *tree_column);
```

Emits the “clicked” signal on the column. This function will only work if `tree_column` is clickable.

Parameters

tree_column	a GtkTreeViewColumn

---

gtk_tree_view_column_set_title ()

```
void
gtk_tree_view_column_set_title (GtkTreeViewColumn *tree_column,
                                const gchar *title);
```

Sets the title of the `tree_column` . If a custom widget has been set, then this value is ignored.

Parameters

tree_column	A GtkTreeViewColumn.
title	The title of the `tree_column` .

---

gtk_tree_view_column_get_title ()

```
const gchar *
gtk_tree_view_column_get_title (GtkTreeViewColumn *tree_column);
```

Returns the title of the widget.

Parameters

tree_column	A GtkTreeViewColumn.

Returns

the title of the column. This string should not be modified or freed.

---

gtk_tree_view_column_set_expand ()

```
void
gtk_tree_view_column_set_expand (GtkTreeViewColumn *tree_column,
                                 gboolean expand);
```

Sets the column to take available extra space. This space is shared equally amongst all columns that have the expand set to `TRUE`. If no column has this option set, then the last column gets all extra space. By default, every column is created with this `FALSE`.

Along with “fixed-width”, the “expand” property changes when the column is resized by the user.

Parameters

tree_column	A GtkTreeViewColumn.
expand	`TRUE` if the column should expand to fill available space.

Since: 2.4

---

gtk_tree_view_column_get_expand ()

```
gboolean
gtk_tree_view_column_get_expand (GtkTreeViewColumn *tree_column);
```

Returns `TRUE` if the column expands to fill available space.

Parameters

tree_column	A GtkTreeViewColumn.

Returns

`TRUE` if the column expands to fill available space.

Since: 2.4

---

gtk_tree_view_column_set_clickable ()

```
void
gtk_tree_view_column_set_clickable (GtkTreeViewColumn *tree_column,
                                    gboolean clickable);
```

Sets the header to be active if `clickable` is `TRUE`. When the header is active, then it can take keyboard focus, and can be clicked.

Parameters

tree_column	A GtkTreeViewColumn.	
clickable	`TRUE` if the header is active.	

---

gtk_tree_view_column_get_clickable ()

```
gboolean
gtk_tree_view_column_get_clickable (GtkTreeViewColumn *tree_column);
```

Returns `TRUE` if the user can click on the header for the column.

Parameters

tree_column	a GtkTreeViewColumn	

Returns

`TRUE` if user can click the column header.

---

gtk_tree_view_column_set_widget ()

```
void
gtk_tree_view_column_set_widget (GtkTreeViewColumn *tree_column,
                                 GtkWidget *widget);
```

Sets the widget in the header to be `widget` . If widget is `NULL`, then the header button is set with a GtkLabel set to the title of `tree_column` .

Parameters

tree_column	A GtkTreeViewColumn.	
widget	A child GtkWidget, or `NULL`.	[allow-none]

---

gtk_tree_view_column_get_widget ()

```
GtkWidget *
gtk_tree_view_column_get_widget (GtkTreeViewColumn *tree_column);
```

Returns the GtkWidget in the button on the column header. If a custom widget has not been set then NULL is returned.

Parameters

tree_column	A GtkTreeViewColumn.

Returns

The GtkWidget in the column header, or NULL.

[nullable][transfer none]

---

gtk_tree_view_column_get_button ()

```
GtkWidget *
gtk_tree_view_column_get_button (GtkTreeViewColumn *tree_column);
```

Returns the button used in the treeview column header

Parameters

tree_column	A GtkTreeViewColumn

Returns

The button for the column header.

[transfer none]

Since: 3.0

---

gtk_tree_view_column_set_alignment ()

```
void
gtk_tree_view_column_set_alignment (GtkTreeViewColumn *tree_column,
                                    gfloat xalign);
```

Sets the alignment of the title or custom widget inside the column header. The alignment determines its location inside the button -- 0.0 for left, 0.5 for center, 1.0 for right.

Parameters

tree_column	A GtkTreeViewColumn.
xalign	The alignment, which is between [0.0 and 1.0] inclusive.

---

gtk_tree_view_column_get_alignment ()

```
gfloat
gtk_tree_view_column_get_alignment (GtkTreeViewColumn *tree_column);
```

Returns the current x alignment of `tree_column` . This value can range between 0.0 and 1.0.

Parameters

tree_column	A GtkTreeViewColumn.

Returns

The current alignent of `tree_column` .

---

gtk_tree_view_column_set_reorderable ()

```
void
gtk_tree_view_column_set_reorderable (GtkTreeViewColumn *tree_column,
                                      gboolean reorderable);
```

If `reorderable` is `TRUE`, then the column can be reordered by the end user dragging the header.

Parameters

tree_column	A GtkTreeViewColumn
reorderable	`TRUE`, if the column can be reordered.

---

gtk_tree_view_column_get_reorderable ()

```
gboolean
gtk_tree_view_column_get_reorderable (GtkTreeViewColumn *tree_column);
```

Returns `TRUE` if the `tree_column` can be reordered by the user.

Parameters

tree_column	A GtkTreeViewColumn

Returns

`TRUE` if the `tree_column` can be reordered by the user.

---

gtk_tree_view_column_set_sort_column_id ()

```
void
```

```
gtk_tree_view_column_set_sort_column_id
                                (GtkTreeViewColumn *tree_column,
                                 gint sort_column_id);
```

Sets the logical `sort_column_id` that this column sorts on when this column is selected for sorting. Doing so makes the column header clickable.

Parameters

tree_column	a GtkTreeViewColumn
sort_column_id	The `sort_column_id` of the model to sort on.

---

gtk_tree_view_column_get_sort_column_id ()

```
gint
gtk_tree_view_column_get_sort_column_id
                                (GtkTreeViewColumn *tree_column);
```

Gets the logical `sort_column_id` that the model sorts on when this column is selected for sorting. See `gtk_tree_view_column_set_sort_column_id()`.

Parameters

tree_column	a GtkTreeViewColumn

Returns

the current `sort_column_id` for this column, or -1 if this column can't be used for sorting.

---

gtk_tree_view_column_set_sort_indicator ()

```
void
gtk_tree_view_column_set_sort_indicator
                                (GtkTreeViewColumn *tree_column,
                                 gboolean setting);
```

Call this function with a `setting` of `TRUE` to display an arrow in the header button indicating the column is sorted. Call `gtk_tree_view_column_set_sort_order()` to change the direction of the arrow.

Parameters

tree_column	a GtkTreeViewColumn
setting	`TRUE` to display an indicator that the column is sorted

---

gtk_tree_view_column_get_sort_indicator ()

```
gboolean
gtk_tree_view_column_get_sort_indicator
                                (GtkTreeViewColumn *tree_column);
```

Gets the value set by gtk_tree_view_column_set_sort_indicator().

Parameters

tree_column	a GtkTreeViewColumn

Returns

whether the sort indicator arrow is displayed

---

gtk_tree_view_column_set_sort_order ()

```
void
gtk_tree_view_column_set_sort_order (GtkTreeViewColumn *tree_column,
                                     GtkSortType order);
```

Changes the appearance of the sort indicator.

This does not actually sort the model. Use gtk_tree_view_column_set_sort_column_id() if you want automatic sorting support. This function is primarily for custom sorting behavior, and should be used in conjunction with gtk_tree_sortable_set_sort_column_id() to do that. For custom models, the mechanism will vary.

The sort indicator changes direction to indicate normal sort or reverse sort. Note that you must have the sort indicator enabled to see anything when calling this function; see gtk_tree_view_column_set_sort_indicator().

Parameters

tree_column	a GtkTreeViewColumn
order	sort order that the sort indicator should indicate

---

gtk_tree_view_column_get_sort_order ()

```
GtkSortType
gtk_tree_view_column_get_sort_order (GtkTreeViewColumn *tree_column);
```

Gets the value set by gtk_tree_view_column_set_sort_order().

Parameters

tree_column	a GtkTreeViewColumn

Returns

the sort order the sort indicator is indicating

---

gtk_tree_view_column_cell_set_cell_data ()

```
void
gtk_tree_view_column_cell_set_cell_data
                                (GtkTreeViewColumn *tree_column,
                                 GtkTreeModel *tree_model,
                                 GtkTreeIter *iter,
                                 gboolean is_expander,
                                 gboolean is_expanded);
```

Sets the cell renderer based on the `tree_model` and `iter` . That is, for every attribute mapping in `tree_column` , it will get a value from the set column on the `iter` , and use that value to set the attribute on the cell renderer. This is used primarily by the GtkTreeView.

Parameters

tree_column	A GtkTreeViewColumn.
tree_model	The GtkTreeModel to to get the cell renderers attributes from.
iter	The GtkTreeIter to to get the cell renderer's attributes from.
is_expander	`TRUE`, if the row has children
is_expanded	`TRUE`, if the row has visible children

---

gtk_tree_view_column_cell_get_size ()

```
void
gtk_tree_view_column_cell_get_size (GtkTreeViewColumn *tree_column,
                                    const GdkRectangle *cell_area,
                                    gint *x_offset,
                                    gint *y_offset,
                                    gint *width,
                                    gint *height);
```

Obtains the width and height needed to render the column. This is used primarily by the GtkTreeView.

Parameters

tree_column	A GtkTreeViewColumn.	
cell_area	The area a cell in the column will be allocated, or `NULL`.	[allow-none]
x_offset	location to return x offset of a cell relative to `cell_area` , or `NULL`.	[out][optional]
y_offset	location to return y offset of a cell relative to `cell_area` , or `NULL`.	[out][optional]
width	location to return width needed to render a cell, or `NULL`.	[out][optional]

height	location to return height needed to render a cell, or NULL.	[out][optional]

gtk_tree_view_column_cell_get_position ()

```
gboolean
gtk_tree_view_column_cell_get_position
                                (GtkTreeViewColumn *tree_column,
                                 GtkCellRenderer *cell_renderer,
                                 gint *x_offset,
                                 gint *width);
```

Obtains the horizontal position and size of a cell in a column. If the cell is not found in the column, `start_pos` and `width` are not changed and `FALSE` is returned.

Parameters

tree_column	a GtkTreeViewColumn	
cell_renderer	a GtkCellRenderer	
x_offset	return location for the horizontal position of `cell` within `tree_column` , may be `NULL`.	[out][allow-none]
width	return location for the width of `cell` , may be `NULL`.	[out][allow-none]

Returns

`TRUE` if `cell` belongs to `tree_column` .

gtk_tree_view_column_cell_is_visible ()

```
gboolean
gtk_tree_view_column_cell_is_visible (GtkTreeViewColumn *tree_column);
```

Returns `TRUE` if any of the cells packed into the `tree_column` are visible. For this to be meaningful, you must first initialize the cells with `gtk_tree_view_column_cell_set_cell_data()`

Parameters

tree_column	A GtkTreeViewColumn

Returns

`TRUE`, if any of the cells packed into the `tree_column` are currently visible

gtk_tree_view_column_focus_cell ()

```
void
gtk_tree_view_column_focus_cell (GtkTreeViewColumn *tree_column,
                                 GtkCellRenderer *cell);
```

Sets the current keyboard focus to be at `cell` , if the column contains 2 or more editable and activatable cells.

Parameters

tree_column	A GtkTreeViewColumn
cell	A GtkCellRenderer

Since: 2.2

---

gtk_tree_view_column_queue_resize ()

```
void
gtk_tree_view_column_queue_resize (GtkTreeViewColumn *tree_column);
```

Flags the column, and the cell renderers added to this column, to have their sizes renegotiated.

Parameters

tree_column	A GtkTreeViewColumn

Since: 2.8

---

gtk_tree_view_column_get_tree_view ()

```
GtkWidget *
gtk_tree_view_column_get_tree_view (GtkTreeViewColumn *tree_column);
```

Returns the GtkTreeView wherein `tree_column` has been inserted. If `column` is currently not inserted in any tree view, `NULL` is returned.

Parameters

tree_column	A GtkTreeViewColumn

Returns

The tree view wherein `column` has been inserted if any, `NULL` otherwise.

[nullable][transfer none]

Since: 2.12

---

gtk_tree_view_column_get_x_offset ()

```
gint
gtk_tree_view_column_get_x_offset (GtkTreeViewColumn *tree_column);
```

Returns the current X offset of `tree_column` in pixels.

Parameters

tree_column	A GtkTreeViewColumn.

Returns

The current X offset of `tree_column` .

Since: 3.2

Types and Values

enum GtkTreeViewColumnSizing

The sizing method the column uses to determine its width. Please note that `GTK_TREE_VIEW_COLUMN_AUTOSIZE` are inefficient for large views, and can make columns appear choppy.

Members

GTK_TREE_VIEW_COLUMN_GROW_ONLY	Columns only get bigger in reaction to changes in the model
GTK_TREE_VIEW_COLUMN_AUTOSIZE	Columns resize to be the optimal size everytime the model changes.
GTK_TREE_VIEW_COLUMN_FIXED	Columns are a fixed numbers of pixels wide.

---

struct GtkTreeViewColumn

```
struct GtkTreeViewColumn;
```

Property Details

The `“alignment”` property

```
  “alignment”                gfloat
```

X Alignment of the column header text or widget.

Flags: Read / Write

Allowed values: [0,1]

Default value: 0

The “cell-area” property

“cell-area” GtkCellArea *

The GtkCellArea used to layout cell renderers for this column.

If no area is specified when creating the tree view column with gtk_tree_view_column_new_with_area() a horizontally oriented GtkCellAreaBox will be used.

Flags: Read / Write / Construct Only

Since: 3.0

The “clickable” property

“clickable” gboolean

Whether the header can be clicked.

Flags: Read / Write

Default value: FALSE

The “expand” property

“expand” gboolean

Column gets share of extra width allocated to the widget.

Flags: Read / Write

Default value: FALSE

The “fixed-width” property

“fixed-width” gint

Current fixed width of the column.

Flags: Read / Write

Allowed values: >= -1

Default value: -1

The “max-width” property

“max-width” gint

Maximum allowed width of the column.

Flags: Read / Write

Allowed values: >= -1

Default value: -1

The “min-width” property

“min-width” gint

Minimum allowed width of the column.

Flags: Read / Write

Allowed values: >= -1

Default value: -1

The “reorderable” property

“reorderable” gboolean

Whether the column can be reordered around the headers.

Flags: Read / Write

Default value: FALSE

The “resizable” property

“resizable” gboolean

Column is user-resizable.

Flags: Read / Write

Default value: FALSE

The “sizing” property

“sizing” GtkTreeViewColumnSizing

Resize mode of the column.

Flags: Read / Write

Default value: GTK_TREE_VIEW_COLUMN_GROW_ONLY

The “sort-column-id” property

“sort-column-id” gint

Logical sort column ID this column sorts on when selected for sorting. Setting the sort column ID makes the column header clickable. Set to -1 to make the column unsortable.

Flags: Read / Write

Allowed values: >= -1

Default value: -1

Since: 2.18

The “sort-indicator” property

“sort-indicator” gboolean

Whether to show a sort indicator.

Flags: Read / Write

Default value: FALSE

The “sort-order” property

“sort-order” GtkSortType

Sort direction the sort indicator should indicate.

Flags: Read / Write

Default value: GTK_SORT_ASCENDING

The “spacing” property

“spacing” gint

Space which is inserted between cells.

Flags: Read / Write

Allowed values: >= 0

Default value: 0

The “title” property

“title” gchar *

Title to appear in column header.

Flags: Read / Write

Default value: ""

The “visible” property

“visible” gboolean

Whether to display the column.

Flags: Read / Write

Default value: TRUE

The “widget” property

“widget” GtkWidget *

Widget to put in column header button instead of column title.

Flags: Read / Write

The “width” property

“width” gint

Current width of the column.

Flags: Read

Allowed values: >= 0

Default value: 0

The “x-offset” property

“x-offset” gint

Current X position of the column.

Flags: Read

Allowed values: >= -2147483647

Default value: 0

Signal Details

The "clicked" signal

```
void
user_function (GtkTreeViewColumn *treeviewcolumn,
               gpointer           user_data)
```

Flags: Run Last

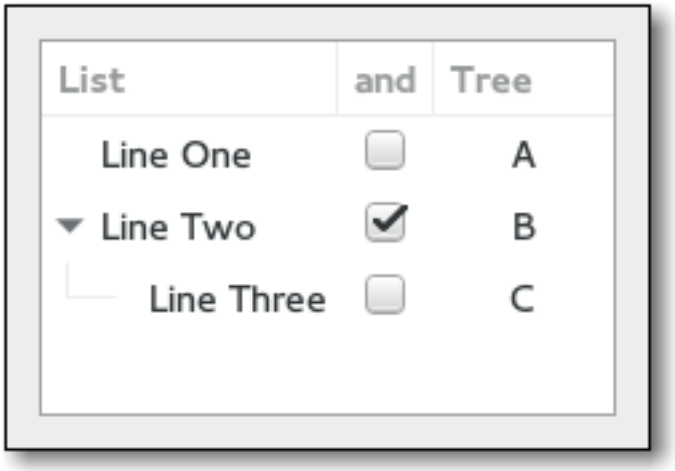
See Also

GtkTreeView, GtkTreeSelection, GtkTreeModel, GtkTreeSortable, GtkTreeModelSort, GtkListStore, GtkTreeStore, GtkCellRenderer, GtkCellEditable, GtkCellRendererPixbuf, GtkCellRendererText, GtkCellRendererToggle, GtkTreeView drag-and-drop

---

GtkTreeView

GtkTreeView — A widget for displaying both trees and lists



Functions

gboolean	(*GtkTreeViewColumnDropFunc) ()
void	(*GtkTreeViewMappingFunc) ()
gboolean	(*GtkTreeViewSearchEqualFunc) ()
GtkWidget *	gtk_tree_view_new ()
gint	gtk_tree_view_get_level_indentation ()
gboolean	gtk_tree_view_get_show_expanders ()
void	gtk_tree_view_set_level_indentation ()
void	gtk_tree_view_set_show_expanders ()
GtkWidget *	gtk_tree_view_new_with_model ()
GtkTreeModel *	gtk_tree_view_get_model ()
void	gtk_tree_view_set_model ()
GtkTreeSelection *	gtk_tree_view_get_selection ()
GtkAdjustment *	gtk_tree_view_get_hadjustment ()
void	gtk_tree_view_set_hadjustment ()
GtkAdjustment *	gtk_tree_view_get_vadjustment ()
void	gtk_tree_view_set_vadjustment ()
gboolean	gtk_tree_view_get_headers_visible ()
void	gtk_tree_view_set_headers_visible ()
void	gtk_tree_view_columns_autosize ()
gboolean	gtk_tree_view_get_headers_clickable ()

void	gtk_tree_view_set_headers_clickable ()
void	gtk_tree_view_set_rules_hint ()
gboolean	gtk_tree_view_get_rules_hint ()
void	gtk_tree_view_set_activate_on_single_click ()
gboolean	gtk_tree_view_get_activate_on_single_click ()
gint	gtk_tree_view_append_column ()
gint	gtk_tree_view_remove_column ()
gint	gtk_tree_view_insert_column ()
gint	gtk_tree_view_insert_column_with_attributes ()
gint	gtk_tree_view_insert_column_with_data_func ()
guint	gtk_tree_view_get_n_columns ()
GtkTreeViewColumn *	gtk_tree_view_get_column ()
GList *	gtk_tree_view_get_columns ()
void	gtk_tree_view_move_column_after ()
void	gtk_tree_view_set_expander_column ()
GtkTreeViewColumn *	gtk_tree_view_get_expander_column ()
void	gtk_tree_view_set_column_drag_function ()
void	gtk_tree_view_scroll_to_point ()
void	gtk_tree_view_scroll_to_cell ()
void	gtk_tree_view_set_cursor ()
void	gtk_tree_view_set_cursor_on_cell ()
void	gtk_tree_view_get_cursor ()
void	gtk_tree_view_row_activated ()
void	gtk_tree_view_expand_all ()
void	gtk_tree_view_collapse_all ()
void	gtk_tree_view_expand_to_path ()
gboolean	gtk_tree_view_expand_row ()
gboolean	gtk_tree_view_collapse_row ()
void	gtk_tree_view_map_expanded_rows ()
gboolean	gtk_tree_view_row_expanded ()
void	gtk_tree_view_set_reorderable ()
gboolean	gtk_tree_view_get_reorderable ()
gboolean	gtk_tree_view_get_path_at_pos ()
gboolean	gtk_tree_view_is_blank_at_pos ()
void	gtk_tree_view_get_cell_area ()
void	gtk_tree_view_get_background_area ()
void	gtk_tree_view_get_visible_rect ()
gboolean	gtk_tree_view_get_visible_range ()
GdkWindow *	gtk_tree_view_get_bin_window ()
void	gtk_tree_view_convert_bin_window_to_tree_coords ()
void	gtk_tree_view_convert_bin_window_to_widget_coords ()
void	gtk_tree_view_convert_tree_to_bin_window_coords ()
void	gtk_tree_view_convert_tree_to_widget_coords ()
void	gtk_tree_view_convert_widget_to_bin_window_coords ()
void	gtk_tree_view_convert_widget_to_tree_coords ()
void	gtk_tree_view_enable_model_drag_dest ()
void	gtk_tree_view_enable_model_drag_source ()
void	gtk_tree_view_unset_rows_drag_source ()
void	gtk_tree_view_unset_rows_drag_dest ()
void	gtk_tree_view_set_drag_dest_row ()

void	gtk_tree_view_get_drag_dest_row ()
gboolean	gtk_tree_view_get_dest_row_at_pos ()
cairo_surface_t *	gtk_tree_view_create_row_drag_icon ()
void	gtk_tree_view_set_enable_search ()
gboolean	gtk_tree_view_get_enable_search ()
gint	gtk_tree_view_get_search_column ()
void	gtk_tree_view_set_search_column ()
GtkTreeViewSearchEqualFunc	gtk_tree_view_get_search_equal_func ()
void	gtk_tree_view_set_search_equal_func ()
GtkEntry *	gtk_tree_view_get_search_entry ()
void	gtk_tree_view_set_search_entry ()
void	(*GtkTreeViewSearchPositionFunc) ()
GtkTreeViewSearchPositionFunc	gtk_tree_view_get_search_position_func ()
void	gtk_tree_view_set_search_position_func ()
gboolean	gtk_tree_view_get_fixed_height_mode ()
void	gtk_tree_view_set_fixed_height_mode ()
gboolean	gtk_tree_view_get_hover_selection ()
void	gtk_tree_view_set_hover_selection ()
gboolean	gtk_tree_view_get_hover_expand ()
void	gtk_tree_view_set_hover_expand ()
void	(*GtkTreeDestroyCountFunc) ()
void	gtk_tree_view_set_destroy_count_func ()
gboolean	(*GtkTreeViewRowSeparatorFunc) ()
GtkTreeViewRowSeparatorFunc	gtk_tree_view_get_row_separator_func ()
void	gtk_tree_view_set_row_separator_func ()
gboolean	gtk_tree_view_get_rubber_banding ()
void	gtk_tree_view_set_rubber_banding ()
gboolean	gtk_tree_view_is_rubber_banding_active ()
gboolean	gtk_tree_view_get_enable_tree_lines ()
void	gtk_tree_view_set_enable_tree_lines ()
GtkTreeViewGridLines	gtk_tree_view_get_grid_lines ()
void	gtk_tree_view_set_grid_lines ()
void	gtk_tree_view_set_tooltip_row ()
void	gtk_tree_view_set_tooltip_cell ()
gboolean	gtk_tree_view_get_tooltip_context ()
gint	gtk_tree_view_get_tooltip_column ()
void	gtk_tree_view_set_tooltip_column ()

Properties

gboolean	activate-on-single-click	Read / Write
GtkTreeViewGridLines	enable-grid-lines	Read / Write
gboolean	enable-search	Read / Write
gboolean	enable-tree-lines	Read / Write
GtkTreeViewColumn *	expander-column	Read / Write
gboolean	fixed-height-mode	Read / Write
gboolean	headers-clickable	Read / Write
gboolean	headers-visible	Read / Write
gboolean	hover-expand	Read / Write
gboolean	hover-selection	Read / Write
gint	level-indentation	Read / Write

GtkTreeModel *	model	Read / Write
gboolean	reorderable	Read / Write
gboolean	rubber-banding	Read / Write
gboolean	rules-hint	Read / Write
gint	search-column	Read / Write
gboolean	show-expanders	Read / Write
gint	tooltip-column	Read / Write

Style Properties

gboolean	allow-rules	Read
GdkColor *	even-row-color	Read
gint	expander-size	Read
gchar *	grid-line-pattern	Read
gint	grid-line-width	Read
gint	horizontal-separator	Read
gboolean	indent-expanders	Read
GdkColor *	odd-row-color	Read
gchar *	tree-line-pattern	Read
gint	tree-line-width	Read
gint	vertical-separator	Read

Signals

void	columns-changed	Run Last
void	cursor-changed	Run Last
gboolean	expand-collapse-cursor-row	Action
gboolean	move-cursor	Action
void	row-activated	Action
void	row-collapsed	Run Last
void	row-expanded	Run Last
gboolean	select-all	Action
gboolean	select-cursor-parent	Action
gboolean	select-cursor-row	Action
gboolean	start-interactive-search	Action
gboolean	test-collapse-row	Run Last
gboolean	test-expand-row	Run Last
gboolean	toggle-cursor-row	Action
gboolean	unselect-all	Action

Types and Values

struct	GtkTreeView
enum	GtkTreeViewDropPosition
	GtkTreeViewPrivate
enum	GtkTreeViewGridLines

Object Hierarchy

```
GObject
```

```
    └── GInitiallyUnowned
        └── GtkWidget
            └── GtkContainer
                └── GtkTreeView
```

Implemented Interfaces

GtkTreeView implements AtkImplementorIface, GtkBuildable and GtkScrollable.

Includes

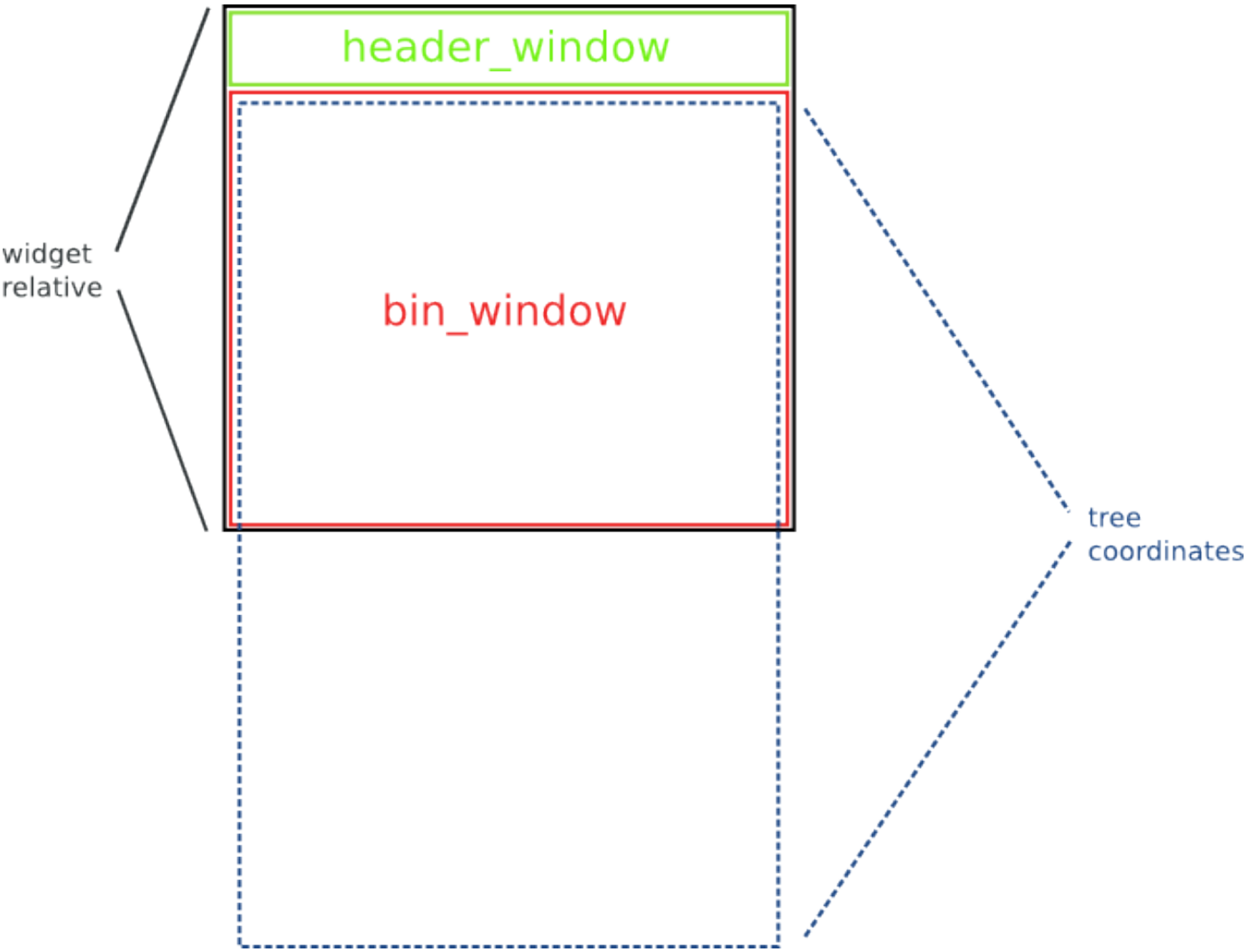
```
#include <gtk/gtk.h>
```

Description

Widget that displays any object that implements the GtkTreeModel interface.

Please refer to the tree widget conceptual overview for an overview of all the objects and data types related to the tree widget and how they work together.

Several different coordinate systems are exposed in the GtkTreeView API. These are:



Coordinate systems in GtkTreeView API:

- Widget coordinates: Coordinates relative to the widget (usually `widget->window`).
- Bin window coordinates: Coordinates relative to the window that GtkTreeView renders to.
- Tree coordinates: Coordinates relative to the entire scrollable area of GtkTreeView. These coordinates start at (0, 0) for row 0 of the tree.

Several functions are available for converting between the different coordinate systems. The most common translations are between widget and bin window coordinates and between bin window and tree coordinates. For the former you can use `gtk_tree_view_convert_widget_to_bin_window_coords()` (and vice versa), for the latter `gtk_tree_view_convert_bin_window_to_tree_coords()` (and vice versa).

GtkTreeView as GtkBuildable

The GtkTreeView implementation of the GtkBuildable interface accepts GtkTreeViewColumn objects as <child> elements and exposes the internal GtkTreeSelection in UI definitions.

An example of a UI definition fragment with GtkTreeView:

```
<object class="GtkTreeView" id="treeview">
  <property
name="model">liststore1</property>
  <child>
    <object class="GtkTreeViewColumn"
id="test-column">
      <property name="title">Test</property>
      <child>
        <object class="GtkCellRendererText"
id="test-renderer"/>
        <attributes>
          <attribute
name="text">1</attribute>
        </attributes>
      </child>
    </object>
  </child>
  <child internal-child="selection">
    <object class="GtkTreeSelection"
id="selection">
      <signal name="changed"
handler="on_treeview_selection_changed"/>
    </object>
  </child>
</object>
```

CSS nodes

```
treeview.view
├── header
│   ├── <column header>
┊   ┊
│   └── <column header>
│
└── [rubberband]
```

GtkTreeView has a main CSS node with name treeview and style class .view. It has a subnode with name header, which is the parent for all the column header widgets' CSS nodes. For rubberband selection, a subnode with name rubberband is used.

Functions

GtkTreeViewColumnDropFunc ()

```
gboolean
(*GtkTreeViewColumnDropFunc) (GtkTreeView *tree_view,
                              GtkTreeViewColumn *column,
                              GtkTreeViewColumn *prev_column,
                              GtkTreeViewColumn *next_column,
                              gpointer data);
```

Function type for determining whether `column` can be dropped in a particular spot (as determined by `prev_column` and `next_column`). In left to right locales, `prev_column` is on the left of the potential drop spot, and `next_column` is on the right. In right to left mode, this is reversed. This function should return `TRUE` if the spot is a valid drop spot. Please note that returning `TRUE` does not actually indicate that the column drop was made, but is meant only to indicate a possible drop spot to the user.

Parameters

tree_view	A GtkTreeView	
column	The GtkTreeViewColumn being dragged	
prev_column	A GtkTreeViewColumn on one side of `column`	
next_column	A GtkTreeViewColumn on the other side of `column`	
data	user data.	[closure]

Returns

`TRUE`, if `column` can be dropped in this spot

GtkTreeViewMappingFunc ()

```
void
(*GtkTreeViewMappingFunc) (GtkTreeView *tree_view,
                           GtkTreePath *path,
                           gpointer user_data);
```

Function used for `gtk_tree_view_map_expanded_rows()`.

Parameters

tree_view	A GtkTreeView
path	The path that’s expanded
user_data	user data

GtkTreeViewSearchEqualFunc ()

```
gboolean
(*GtkTreeViewSearchEqualFunc) (GtkTreeModel *model,
                               gint column,
                               const gchar *key,
                               GtkTreeIter *iter,
                               gpointer search_data);
```

A function used for checking whether a row in `model` matches a search key string entered by the user. Note the return value is reversed from what you would normally expect, though it has some similarity to `strcmp()` returning 0 for equal strings.

Parameters

model	the GtkTreeModel being searched	
column	the search column set by `gtk_tree_view_set_search_column()`	
key	the key string to compare with	
iter	a GtkTreeIter pointing the row of `model` that should be compared with `key`.	
search_data	user data from `gtk_tree_view_set_search_equal_func()`.	[closure]

Returns

`FALSE` if the row matches, `TRUE` otherwise.

---

gtk_tree_view_new ()

```
GtkWidget *
gtk_tree_view_new (void);
```

Creates a new GtkTreeView widget.

Returns

A newly created GtkTreeView widget.

---

gtk_tree_view_get_level_indentation ()

```
gint
gtk_tree_view_get_level_indentation (GtkTreeView *tree_view);
```

Returns the amount, in pixels, of extra indentation for child levels in `tree_view`.

Parameters

tree_view	a GtkTreeView.

Returns

the amount of extra indentation for child levels in `tree_view`. A return value of 0 means that this feature is disabled.

Since: 2.12

---

gtk_tree_view_get_show_expanders ()

```
gboolean
gtk_tree_view_get_show_expanders (GtkTreeView *tree_view);
```

Returns whether or not expanders are drawn in `tree_view`.

Parameters

tree_view	a GtkTreeView.

Returns

`TRUE` if expanders are drawn in `tree_view`, `FALSE` otherwise.

Since: 2.12

---

gtk_tree_view_set_level_indentation ()

```
void
gtk_tree_view_set_level_indentation (GtkTreeView *tree_view,
                                     gint indentation);
```

Sets the amount of extra indentation for child levels to use in `tree_view` in addition to the default indentation. The value should be specified in pixels, a value of 0 disables this feature and in this case only the default indentation will be used. This does not have any visible effects for lists.

Parameters

tree_view	a GtkTreeView
indentation	the amount, in pixels, of extra indentation in `tree_view`.

Since: 2.12

---

gtk_tree_view_set_show_expanders ()

```
void
gtk_tree_view_set_show_expanders (GtkTreeView *tree_view,
                                  gboolean enabled);
```

Sets whether to draw and enable expanders and indent child rows in `tree_view`. When disabled there will be no expanders visible in trees and there will be no way to expand and collapse rows by default. Also note that hiding the expanders will disable the default indentation. You can set a custom indentation in this case using `gtk_tree_view_set_level_indentation()`. This does not have any visible effects for lists.

Parameters

tree_view	a GtkTreeView
enabled	`TRUE` to enable expander drawing, `FALSE` otherwise.

Since: 2.12

---

gtk_tree_view_new_with_model ()

```
GtkWidget *
gtk_tree_view_new_with_model (GtkTreeModel *model);
```

Creates a new GtkTreeView widget with the model initialized to `model`.

Parameters

model	the model.

Returns

A newly created GtkTreeView widget.

---

gtk_tree_view_get_model ()

```
GtkTreeModel *
gtk_tree_view_get_model (GtkTreeView *tree_view);
```

Returns the model the GtkTreeView is based on. Returns `NULL` if the model is unset.

Parameters

tree_view	a GtkTreeView

Returns

A GtkTreeModel, or `NULL` if none is currently being used.

[transfer none][nullable]

gtk_tree_view_set_model ()

```
void
gtk_tree_view_set_model (GtkTreeView *tree_view,
                         GtkTreeModel *model);
```

Sets the model for a GtkTreeView. If the `tree_view` already has a model set, it will remove it before setting the new model. If `model` is `NULL`, then it will unset the old model.

Parameters

tree_view	A GtkTreeView.	
model	The model.	[allow-none]

---

gtk_tree_view_get_selection ()

```
GtkTreeSelection *
gtk_tree_view_get_selection (GtkTreeView *tree_view);
```

Gets the GtkTreeSelection associated with `tree_view`.

Parameters

tree_view	A GtkTreeView.

Returns

A GtkTreeSelection object.

[transfer none]

---

gtk_tree_view_get_hadjustment ()

```
GtkAdjustment *
gtk_tree_view_get_hadjustment (GtkTreeView *tree_view);
```

`gtk_tree_view_get_hadjustment` has been deprecated since version 3.0 and should not be used in newly-written code.

Use `gtk_scrollable_get_hadjustment()`

Gets the GtkAdjustment currently being used for the horizontal aspect.

Parameters

tree_view	A GtkTreeView

Returns

A GtkAdjustment object, or `NULL` if none is currently being used.

[transfer none]

---

gtk_tree_view_set_hadjustment ()

```
void
gtk_tree_view_set_hadjustment (GtkTreeView *tree_view,
                               GtkAdjustment *adjustment);
```

`gtk_tree_view_set_hadjustment` has been deprecated since version 3.0 and should not be used in newly-written code.

Use `gtk_scrollable_set_hadjustment()`

Sets the GtkAdjustment for the current horizontal aspect.

Parameters

tree_view	A GtkTreeView
adjustment	The GtkAdjustment to set, or `NULL`. [allow-none]

---

gtk_tree_view_get_vadjustment ()

```
GtkAdjustment *
gtk_tree_view_get_vadjustment (GtkTreeView *tree_view);
```

`gtk_tree_view_get_vadjustment` has been deprecated since version 3.0 and should not be used in newly-written code.

Use `gtk_scrollable_get_vadjustment()`

Gets the GtkAdjustment currently being used for the vertical aspect.

Parameters

tree_view	A GtkTreeView

Returns

A GtkAdjustment object, or `NULL` if none is currently being used.

[transfer none]

---

gtk_tree_view_set_vadjustment ()

```
void
```

```
gtk_tree_view_set_vadjustment (GtkTreeView *tree_view,
                               GtkAdjustment *adjustment);
```

`gtk_tree_view_set_vadjustment` has been deprecated since version 3.0 and should not be used in newly-written code.

Use `gtk_scrollable_set_vadjustment()`

Sets the GtkAdjustment for the current vertical aspect.

Parameters

tree_view	A GtkTreeView
adjustment	The GtkAdjustment to set, or `NULL`. [allow-none]

---

gtk_tree_view_get_headers_visible ()

```
gboolean
gtk_tree_view_get_headers_visible (GtkTreeView *tree_view);
```

Returns `TRUE` if the headers on the `tree_view` are visible.

Parameters

tree_view	A GtkTreeView.

Returns

Whether the headers are visible or not.

---

gtk_tree_view_set_headers_visible ()

```
void
gtk_tree_view_set_headers_visible (GtkTreeView *tree_view,
                                   gboolean headers_visible);
```

Sets the visibility state of the headers.

Parameters

tree_view	A GtkTreeView.
headers_visible	`TRUE` if the headers are visible

---

gtk_tree_view_columns_autosize ()

```
void
gtk_tree_view_columns_autosize (GtkTreeView *tree_view);
```

Resizes all columns to their optimal width. Only works after the treeview has been realized.

Parameters

tree_view	A GtkTreeView.

---

gtk_tree_view_get_headers_clickable ()

```
gboolean
gtk_tree_view_get_headers_clickable (GtkTreeView *tree_view);
```

Returns whether all header columns are clickable.

Parameters

tree_view	A GtkTreeView.

Returns

`TRUE` if all header columns are clickable, otherwise `FALSE`

Since: 2.10

---

gtk_tree_view_set_headers_clickable ()

```
void
gtk_tree_view_set_headers_clickable (GtkTreeView *tree_view,
                                     gboolean setting);
```

Allow the column title buttons to be clicked.

Parameters

tree_view	A GtkTreeView.
setting	`TRUE` if the columns are clickable.

---

gtk_tree_view_set_rules_hint ()

```
void
gtk_tree_view_set_rules_hint (GtkTreeView *tree_view,
                              gboolean setting);
```

`gtk_tree_view_set_rules_hint` has been deprecated since version 3.14 and should not be used in newly-written code.

Sets a hint for the theme to draw even/odd rows in the `tree_view` with different colors, also known as "zebra striping".

This function tells the GTK+ theme that the user interface for your application requires users to read across tree

rows and associate cells with one another.

Do not use it just because you prefer the appearance of the ruled tree; that's a question for the theme. Some themes will draw tree rows in alternating colors even when rules are turned off, and users who prefer that appearance all the time can choose those themes. You should call this function only as a semantic hint to the theme engine that your tree makes alternating colors useful from a functional standpoint (since it has lots of columns, generally).

Parameters

tree_view	a GtkTreeView
setting	TRUE if the tree requires reading across rows

---

gtk_tree_view_get_rules_hint ()

```
gboolean
gtk_tree_view_get_rules_hint (GtkTreeView *tree_view);
```

`gtk_tree_view_get_rules_hint` has been deprecated since version 3.14 and should not be used in newly-written code.

Gets the setting set by `gtk_tree_view_set_rules_hint()`.

Parameters

tree_view	a GtkTreeView

Returns

TRUE if the hint is set

---

gtk_tree_view_set_activate_on_single_click ()

```
void
gtk_tree_view_set_activate_on_single_click
                                (GtkTreeView *tree_view,
                                 gboolean single);
```

Cause the "row-activated" signal to be emitted on a single click instead of a double click.

Parameters

tree_view	a GtkTreeView
single	TRUE to emit row-activated on a single click

Since: 3.8

---

gtk_tree_view_get_activate_on_single_click ()

```
gboolean
gtk_tree_view_get_activate_on_single_click
                                (GtkTreeView *tree_view);
```

Gets the setting set by `gtk_tree_view_set_activate_on_single_click()`.

Parameters

tree_view	a GtkTreeView

Returns

`TRUE` if row-activated will be emitted on a single click

Since: 3.8

---

gtk_tree_view_append_column ()

```
gint
gtk_tree_view_append_column (GtkTreeView *tree_view,
                             GtkTreeViewColumn *column);
```

Appends `column` to the list of columns. If `tree_view` has “fixed_height” mode enabled, then `column` must have its “sizing” property set to be GTK_TREE_VIEW_COLUMN_FIXED.

Parameters

tree_view	A GtkTreeView.
column	The GtkTreeViewColumn to add.

Returns

The number of columns in `tree_view` after appending.

---

gtk_tree_view_remove_column ()

```
gint
gtk_tree_view_remove_column (GtkTreeView *tree_view,
                             GtkTreeViewColumn *column);
```

Removes `column` from `tree_view` .

Parameters

tree_view	A GtkTreeView.
column	The GtkTreeViewColumn to

remove.

Returns

The number of columns in `tree_view` after removing.

---

gtk_tree_view_insert_column ()

```
gint
gtk_tree_view_insert_column (GtkTreeView *tree_view,
                             GtkTreeViewColumn *column,
                             gint position);
```

This inserts the `column` into the `tree_view` at `position` . If `position` is -1, then the column is inserted at the end. If `tree_view` has “fixed_height” mode enabled, then `column` must have its “sizing” property set to be GTK_TREE_VIEW_COLUMN_FIXED.

Parameters

tree_view	A GtkTreeView.
column	The GtkTreeViewColumn to be inserted.
position	The position to insert `column` in.

Returns

The number of columns in `tree_view` after insertion.

---

gtk_tree_view_insert_column_with_attributes ()

```
gint
gtk_tree_view_insert_column_with_attributes
                              (GtkTreeView *tree_view,
                               gint position,
                               const gchar *title,
                               GtkCellRenderer *cell,
                               ...);
```

Creates a new GtkTreeViewColumn and inserts it into the `tree_view` at `position` . If `position` is -1, then the newly created column is inserted at the end. The column is initialized with the attributes given. If `tree_view` has “fixed_height” mode enabled, then the new column will have its sizing property set to be GTK_TREE_VIEW_COLUMN_FIXED.

Parameters

tree_view	A GtkTreeView
position	The position to insert the new column in
title	The title to set the header to

cell	The GtkCellRenderer
...	A NULL-terminated list of attributes

Returns

The number of columns in `tree_view` after insertion.

---

gtk_tree_view_insert_column_with_data_func ()

```
gint
gtk_tree_view_insert_column_with_data_func
                                (GtkTreeView *tree_view,
                                 gint position,
                                 const gchar *title,
                                 GtkCellRenderer *cell,
                                 GtkTreeCellDataFunc func,
                                 gpointer data,
                                 GDestroyNotify dnotify);
```

Convenience function that inserts a new column into the GtkTreeView with the given cell renderer and a GtkTreeCellDataFunc to set cell renderer attributes (normally using data from the model). See also `gtk_tree_view_column_set_cell_data_func()`, `gtk_tree_view_column_pack_start()`. If `tree_view` has "fixed_height" mode enabled, then the new column will have its "sizing" property set to be GTK_TREE_VIEW_COLUMN_FIXED.

Parameters

tree_view	a GtkTreeView
position	Position to insert, -1 for append
title	column title
cell	cell renderer for column
func	function to set attributes of cell renderer
data	data for `func`
dnotify	destroy notifier for `data`

Returns

number of columns in the tree view post-insert

---

gtk_tree_view_get_n_columns ()

```
guint
gtk_tree_view_get_n_columns (GtkTreeView *tree_view);
```

Queries the number of columns in the given `tree_view`.

Parameters

tree_view	a GtkTreeView

Returns

The number of columns in the `tree_view`

Since: 3.4

---

gtk_tree_view_get_column ()

```
GtkTreeViewColumn *
gtk_tree_view_get_column (GtkTreeView *tree_view,
                          gint n);
```

Gets the GtkTreeViewColumn at the given position in the tree_view.

Parameters

tree_view	A GtkTreeView.
n	The position of the column, counting from 0.

Returns

The GtkTreeViewColumn, or `NULL` if the position is outside the range of columns.

[nullable][transfer none]

---

gtk_tree_view_get_columns ()

```
GList *
gtk_tree_view_get_columns (GtkTreeView *tree_view);
```

Returns a GList of all the GtkTreeViewColumn s currently in `tree_view` . The returned list must be freed with `g_list_free()`.

Parameters

tree_view	A GtkTreeView

Returns

A list of GtkTreeViewColumn s.

[element-type GtkTreeViewColumn][transfer container]

---

gtk_tree_view_move_column_after ()

```
void
gtk_tree_view_move_column_after (GtkTreeView *tree_view,
                                 GtkTreeViewColumn *column,
                                 GtkTreeViewColumn *base_column);
```

Moves `column` to be after to `base_column` . If `base_column` is `NULL`, then `column` is placed in the first position.

Parameters

tree_view	A GtkTreeView	
column	The GtkTreeViewColumn to be moved.	
base_column	The GtkTreeViewColumn to be moved relative to, or `NULL`.	[allow-none]

---

gtk_tree_view_set_expander_column ()

```
void
gtk_tree_view_set_expander_column (GtkTreeView *tree_view,
                                   GtkTreeViewColumn *column);
```

Sets the column to draw the expander arrow at. It must be in `tree_view` . If `column` is `NULL`, then the expander arrow is always at the first visible column.

If you do not want expander arrow to appear in your tree, set the expander column to a hidden column.

Parameters

tree_view	A GtkTreeView	
column	`NULL`, or the column to draw the expander arrow at.	[nullable]

---

gtk_tree_view_get_expander_column ()

```
GtkTreeViewColumn *
gtk_tree_view_get_expander_column (GtkTreeView *tree_view);
```

Returns the column that is the current expander column. This column has the expander arrow drawn next to it.

Parameters

tree_view	A GtkTreeView

Returns

The expander column.

[transfer none]

---

gtk_tree_view_set_column_drag_function ()

```
void
gtk_tree_view_set_column_drag_function
                                (GtkTreeView *tree_view,
                                 GtkTreeViewColumnDropFunc func,
                                 gpointer user_data,
                                 GDestroyNotify destroy);
```

Sets a user function for determining where a column may be dropped when dragged. This function is called on every column pair in turn at the beginning of a column drag to determine where a drop can take place. The arguments passed to `func` are: the `tree_view` , the GtkTreeViewColumn being dragged, the two GtkTreeViewColumn s determining the drop spot, and `user_data` . If either of the GtkTreeViewColumn arguments for the drop spot are `NULL`, then they indicate an edge. If `func` is set to be `NULL`, then `tree_view` reverts to the default behavior of allowing all columns to be dropped everywhere.

Parameters

tree_view	A GtkTreeView.	
func	A function to determine which columns are reorderable, or `NULL`.	[allow-none]
user_data	User data to be passed to `func` , or `NULL`.	[allow-none]
destroy	Destroy notifier for `user_data` , or `NULL`.	[allow-none]

---

gtk_tree_view_scroll_to_point ()

```
void
gtk_tree_view_scroll_to_point (GtkTreeView *tree_view,
                               gint tree_x,
                               gint tree_y);
```

Scrolls the tree view such that the top-left corner of the visible area is `tree_x` , `tree_y` , where `tree_x` and `tree_y` are specified in tree coordinates. The `tree_view` must be realized before this function is called. If it isn't, you probably want to be using `gtk_tree_view_scroll_to_cell()`.

If either `tree_x` or `tree_y` are -1, then that direction isn’t scrolled.

Parameters

tree_view	a GtkTreeView
tree_x	X coordinate of new top-left pixel of visible area, or -1
tree_y	Y coordinate of new top-left pixel of visible area, or -1

---

gtk_tree_view_scroll_to_cell ()

```
void
gtk_tree_view_scroll_to_cell (GtkTreeView *tree_view,
                              GtkTreePath *path,
                              GtkTreeViewColumn *column,
                              gboolean use_align,
                              gfloat row_align,
                              gfloat col_align);
```

Moves the alignments of `tree_view` to the position specified by `column` and `path` . If `column` is `NULL`, then no horizontal scrolling occurs. Likewise, if `path` is `NULL` no vertical scrolling occurs. At a minimum, one of `column` or `path` need to be non-`NULL`. `row_align` determines where the row is placed, and `col_align` determines where `column` is placed. Both are expected to be between 0.0 and 1.0. 0.0 means left/top alignment, 1.0 means right/bottom alignment, 0.5 means center.

If `use_align` is `FALSE`, then the alignment arguments are ignored, and the tree does the minimum amount of work to scroll the cell onto the screen. This means that the cell will be scrolled to the edge closest to its current position. If the cell is currently visible on the screen, nothing is done.

This function only works if the model is set, and `path` is a valid row on the model. If the model changes before the `tree_view` is realized, the centered path will be modified to reflect this change.

Parameters

tree_view	A GtkTreeView.	
path	The path of the row to move to, or `NULL`.	[allow-none]
column	The GtkTreeViewColumn to move horizontally to, or `NULL`.	[allow-none]
use_align	whether to use alignment arguments, or `FALSE`.	
row_align	The vertical alignment of the row specified by `path` .	
col_align	The horizontal alignment of the column specified by `column` .	

---

gtk_tree_view_set_cursor ()

```
void
gtk_tree_view_set_cursor (GtkTreeView *tree_view,
                          GtkTreePath *path,
                          GtkTreeViewColumn *focus_column,
                          gboolean start_editing);
```

Sets the current keyboard focus to be at `path` , and selects it. This is useful when you want to focus the user's attention on a particular row. If `focus_column` is not `NULL`, then focus is given to the column specified by it. Additionally, if `focus_column` is specified, and `start_editing` is `TRUE`, then editing should be started in the specified cell. This function is often followed by `gtk_widget_grab_focus` (`tree_view`) in order to give keyboard focus to the widget. Please note that editing can only happen when the widget is realized.

If `path` is invalid for `model` , the current cursor (if any) will be unset and the function will return without failing.

Parameters

tree_view	A GtkTreeView	
path	A GtkTreePath	
focus_column	A GtkTreeViewColumn, or NULL.	[allow-none]
start_editing	TRUE if the specified cell should start being edited.	

---

gtk_tree_view_set_cursor_on_cell ()

```
void
gtk_tree_view_set_cursor_on_cell (GtkTreeView *tree_view,
                                  GtkTreePath *path,
                                  GtkTreeViewColumn *focus_column,
                                  GtkCellRenderer *focus_cell,
                                  gboolean start_editing);
```

Sets the current keyboard focus to be at `path` , and selects it. This is useful when you want to focus the user's attention on a particular row. If `focus_column` is not `NULL`, then focus is given to the column specified by it. If `focus_column` and `focus_cell` are not `NULL`, and `focus_column` contains 2 or more editable or activatable cells, then focus is given to the cell specified by `focus_cell` . Additionally, if `focus_column` is specified, and `start_editing` is `TRUE`, then editing should be started in the specified cell. This function is often followed by `gtk_widget_grab_focus` (`tree_view`) in order to give keyboard focus to the widget. Please note that editing can only happen when the widget is realized.

If `path` is invalid for `model` , the current cursor (if any) will be unset and the function will return without failing.

Parameters

tree_view	A GtkTreeView	
path	A GtkTreePath	
focus_column	A GtkTreeViewColumn, or NULL.	[allow-none]
focus_cell	A GtkCellRenderer, or NULL.	[allow-none]
start_editing	TRUE if the specified cell should start being edited.	

Since: 2.2

---

gtk_tree_view_get_cursor ()

```
void
gtk_tree_view_get_cursor (GtkTreeView *tree_view,
                          GtkTreePath **path,
                          GtkTreeViewColumn **focus_column);
```

Fills in `path` and `focus_column` with the current path and focus column. If the cursor isn't currently set, then *`path` will be `NULL`. If no column currently has focus, then *`focus_column` will be `NULL`.

The returned GtkTreePath must be freed with `gtk_tree_path_free()` when you are done with it.

Parameters

tree_view	A GtkTreeView	
path	A pointer to be filled with the current cursor path, or `NULL`.	[out][transfer full][optional][nullable]
focus_column	A pointer to be filled with the current focus column, or `NULL`.	[out][transfer none][optional][nullable]

---

gtk_tree_view_row_activated ()

```
void
gtk_tree_view_row_activated (GtkTreeView *tree_view,
                             GtkTreePath *path,
                             GtkTreeViewColumn *column);
```

Activates the cell determined by `path` and `column`.

Parameters

tree_view	A GtkTreeView
path	The GtkTreePath to be activated.
column	The GtkTreeViewColumn to be activated.

---

gtk_tree_view_expand_all ()

```
void
gtk_tree_view_expand_all (GtkTreeView *tree_view);
```

Recursively expands all nodes in the `tree_view`.

Parameters

tree_view	A GtkTreeView.

---

gtk_tree_view_collapse_all ()

```
void
gtk_tree_view_collapse_all (GtkTreeView *tree_view);
```

Recursively collapses all visible, expanded nodes in `tree_view`.

Parameters

tree_view	A GtkTreeView.

---

gtk_tree_view_expand_to_path ()

```
void
gtk_tree_view_expand_to_path (GtkTreeView *tree_view,
                              GtkTreePath *path);
```

Expands the row at `path` . This will also expand all parent rows of `path` as necessary.

Parameters

tree_view	A GtkTreeView.
path	path to a row.

Since: 2.2

---

gtk_tree_view_expand_row ()

```
gboolean
gtk_tree_view_expand_row (GtkTreeView *tree_view,
                          GtkTreePath *path,
                          gboolean open_all);
```

Opens the row so its children are visible.

Parameters

tree_view	a GtkTreeView
path	path to a row
open_all	whether to recursively expand, or just expand immediate children

Returns

`TRUE` if the row existed and had children

---

gtk_tree_view_collapse_row ()

```
gboolean
gtk_tree_view_collapse_row (GtkTreeView *tree_view,
                            GtkTreePath *path);
```

Collapses a row (hides its child rows, if they exist).

Parameters

tree_view	a GtkTreeView
path	path to a row in the `tree_view`

Returns

TRUE if the row was collapsed.

---

gtk_tree_view_map_expanded_rows ()

```
void
gtk_tree_view_map_expanded_rows (GtkTreeView *tree_view,
                                 GtkTreeViewMappingFunc func,
                                 gpointer data);
```

Calls func on all expanded rows.

Parameters

tree_view	A GtkTreeView	
func	A function to be called.	[scope call]
data	User data to be passed to the function.	

---

gtk_tree_view_row_expanded ()

```
gboolean
gtk_tree_view_row_expanded (GtkTreeView *tree_view,
                            GtkTreePath *path);
```

Returns TRUE if the node pointed to by path is expanded in tree_view.

Parameters

tree_view	A GtkTreeView.
path	A GtkTreePath to test expansion state.

Returns

TRUE if path is expanded.

---

gtk_tree_view_set_reorderable ()

```
void
gtk_tree_view_set_reorderable (GtkTreeView *tree_view,
                               gboolean reorderable);
```

This function is a convenience function to allow you to reorder models that support the GtkTreeDragSourceIface and the GtkTreeDragDestIface. Both GtkTreeStore and GtkListStore support these. If reorderable is TRUE, then the user can reorder the model by dragging and dropping rows. The developer can listen to these changes by connecting to the model's "row-inserted" and "row-deleted" signals. The reordering is implemented by setting up the tree view as a drag source and destination. Therefore, drag and drop can not be

used in a reorderable view for any other purpose.

This function does not give you any degree of control over the order -- any reordering is allowed. If more control is needed, you should probably handle drag and drop manually.

Parameters

tree_view	A GtkTreeView.
reorderable	TRUE, if the tree can be reordered.

---

gtk_tree_view_get_reorderable ()

```
gboolean
gtk_tree_view_get_reorderable (GtkTreeView *tree_view);
```

Retrieves whether the user can reorder the tree via drag-and-drop. See gtk_tree_view_set_reorderable().

Parameters

tree_view	a GtkTreeView

Returns

TRUE if the tree can be reordered.

---

gtk_tree_view_get_path_at_pos ()

```
gboolean
gtk_tree_view_get_path_at_pos (GtkTreeView *tree_view,
                               gint x,
                               gint y,
                               GtkTreePath **path,
                               GtkTreeViewColumn **column,
                               gint *cell_x,
                               gint *cell_y);
```

Finds the path at the point (x , y), relative to bin_window coordinates (please see gtk_tree_view_get_bin_window()). That is, x and y are relative to an events coordinates. x and y must come from an event on the tree_view only where event->window == gtk_tree_view_get_bin_window(). It is primarily for things like popup menus. If path is non-NULL, then it will be filled with the GtkTreePath at that point. This path should be freed with gtk_tree_path_free(). If column is non-NULL, then it will be filled with the column at that point. cell_x and cell_y return the coordinates relative to the cell background (i.e. the background_area passed to gtk_cell_renderer_render()). This function is only meaningful if tree_view is realized. Therefore this function will always return FALSE if tree_view is not realized or does not have a model.

For converting widget coordinates (eg. the ones you get from GtkWidget::query-tooltip), please see gtk_tree_view_convert_widget_to_bin_window_coords().

Parameters

tree_view	A GtkTreeView.	
x	The x position to be identified (relative to bin_window).	
y	The y position to be identified (relative to bin_window).	
path	A pointer to a GtkTreePath pointer to be filled in, or `NULL`.	[out][optional][nullable]
column	A pointer to a GtkTreeViewColumn pointer to be filled in, or `NULL`.	[out][transfer none][optional][nullable]
cell_x	A pointer where the X coordinate relative to the cell can be placed, or `NULL`.	[out][optional]
cell_y	A pointer where the Y coordinate relative to the cell can be placed, or `NULL`.	[out][optional]

Returns

`TRUE` if a row exists at that coordinate.

---

gtk_tree_view_is_blank_at_pos ()

```
gboolean
gtk_tree_view_is_blank_at_pos (GtkTreeView *tree_view,
                               gint x,
                               gint y,
                               GtkTreePath **path,
                               GtkTreeViewColumn **column,
                               gint *cell_x,
                               gint *cell_y);
```

Determine whether the point (`x` , `y`) in `tree_view` is blank, that is no cell content nor an expander arrow is drawn at the location. If so, the location can be considered as the background. You might wish to take special action on clicks on the background, such as clearing a current selection, having a custom context menu or starting rubber banding.

The `x` and `y` coordinate that are provided must be relative to bin_window coordinates. That is, `x` and `y` must come from an event on `tree_view` where `event->window == gtk_tree_view_get_bin_window()`.

For converting widget coordinates (eg. the ones you get from GtkWidget::query-tooltip), please see `gtk_tree_view_convert_widget_to_bin_window_coords()`.

The `path` , `column` , `cell_x` and `cell_y` arguments will be filled in likewise as for `gtk_tree_view_get_path_at_pos()`. Please see `gtk_tree_view_get_path_at_pos()` for more information.

Parameters

tree_view	A GtkTreeView
x	The x position to be identified (relative to bin_window)

y	The y position to be identified (relative to bin_window)	
path	A pointer to a GtkTreePath pointer to be filled in, or NULL.	[out][optional][nullable]
column	A pointer to a GtkTreeViewColumn pointer to be filled in, or NULL.	[out][transfer none][optional][nullable]
cell_x	A pointer where the X coordinate relative to the cell can be placed, or NULL.	[out][optional]
cell_y	A pointer where the Y coordinate relative to the cell can be placed, or NULL.	[out][optional]

Returns

TRUE if the area at the given coordinates is blank, FALSE otherwise.

Since: 3.0

---

gtk_tree_view_get_cell_area ()

```
void
gtk_tree_view_get_cell_area (GtkTreeView *tree_view,
                             GtkTreePath *path,
                             GtkTreeViewColumn *column,
                             GdkRectangle *rect);
```

Fills the bounding rectangle in bin_window coordinates for the cell at the row specified by `path` and the column specified by `column` . If `path` is `NULL`, or points to a path not currently displayed, the `y` and `height` fields of the rectangle will be filled with 0. If `column` is `NULL`, the `x` and `width` fields will be filled with 0. The sum of all cell rects does not cover the entire tree; there are extra pixels in between rows, for example. The returned rectangle is equivalent to the `cell_area` passed to `gtk_cell_renderer_render()`. This function is only valid if `tree_view` is realized.

Parameters

tree_view	a GtkTreeView	
path	a GtkTreePath for the row, or NULL to get only horizontal coordinates.	[allow-none]
column	a GtkTreeViewColumn for the column, or NULL to get only vertical coordinates.	[allow-none]
rect	rectangle to fill with cell rect.	[out]

---

gtk_tree_view_get_background_area ()

```
void
gtk_tree_view_get_background_area (GtkTreeView *tree_view,
                                   GtkTreePath *path,
```

```
                                GtkTreeViewColumn *column,
                                GdkRectangle *rect);
```

Fills the bounding rectangle in bin_window coordinates for the cell at the row specified by `path` and the column specified by `column` . If `path` is `NULL`, or points to a node not found in the tree, the `y` and `height` fields of the rectangle will be filled with 0. If `column` is `NULL`, the `x` and `width` fields will be filled with 0. The returned rectangle is equivalent to the `background_area` passed to `gtk_cell_renderer_render()`. These background areas tile to cover the entire bin window. Contrast with the `cell_area` , returned by `gtk_tree_view_get_cell_area()`, which returns only the cell itself, excluding surrounding borders and the tree expander area.

Parameters

tree_view	a GtkTreeView	
path	a GtkTreePath for the row, or `NULL` to get only horizontal coordinates.	[allow-none]
column	a GtkTreeViewColumn for the column, or `NULL` to get only vertical coordiantes.	[allow-none]
rect	rectangle to fill with cell background rect.	[out]

---

gtk_tree_view_get_visible_rect ()

```
void
gtk_tree_view_get_visible_rect (GtkTreeView *tree_view,
                                GdkRectangle *visible_rect);
```

Fills `visible_rect` with the currently-visible region of the buffer, in tree coordinates. Convert to bin_window coordinates with `gtk_tree_view_convert_tree_to_bin_window_coords()`. Tree coordinates start at 0,0 for row 0 of the tree, and cover the entire scrollable area of the tree.

Parameters

tree_view	a GtkTreeView	
visible_rect	rectangle to fill.	[out]

---

gtk_tree_view_get_visible_range ()

```
gboolean
gtk_tree_view_get_visible_range (GtkTreeView *tree_view,
                                 GtkTreePath **start_path,
                                 GtkTreePath **end_path);
```

Sets `start_path` and `end_path` to be the first and last visible path. Note that there may be invisible paths in between.

The paths should be freed with `gtk_tree_path_free()` after use.

Parameters

tree_view	A GtkTreeView	
start_path	Return location for start of region, or `NULL`.	[out][allow-none]
end_path	Return location for end of region, or `NULL`.	[out][allow-none]

Returns

`TRUE`, if valid paths were placed in `start_path` and `end_path` .

Since: 2.8

---

gtk_tree_view_get_bin_window ()

```
GdkWindow *
gtk_tree_view_get_bin_window (GtkTreeView *tree_view);
```

Returns the window that `tree_view` renders to. This is used primarily to compare to `event->window` to confirm that the event on `tree_view` is on the right window.

Parameters

tree_view	A GtkTreeView

Returns

A GdkWindow, or `NULL` when `tree_view` hasn't been realized yet.

[nullable][transfer none]

---

gtk_tree_view_convert_bin_window_to_tree_coords ()

```
void
gtk_tree_view_convert_bin_window_to_tree_coords
                                (GtkTreeView *tree_view,
                                 gint bx,
                                 gint by,
                                 gint *tx,
                                 gint *ty);
```

Converts bin_window coordinates to coordinates for the tree (the full scrollable area of the tree).

Parameters

tree_view	a GtkTreeView	
bx	X coordinate relative to bin_window	
by	Y coordinate relative to	

	bin_window	
tx	return location for tree X coordinate.	[out]
ty	return location for tree Y coordinate.	[out]

Since: 2.12

---

gtk_tree_view_convert_bin_window_to_widget_coords ()

```
void
gtk_tree_view_convert_bin_window_to_widget_coords
                                (GtkTreeView *tree_view,
                                 gint bx,
                                 gint by,
                                 gint *wx,
                                 gint *wy);
```

Converts bin_window coordinates (see `gtk_tree_view_get_bin_window()`) to widget relative coordinates.

Parameters

tree_view	a GtkTreeView	
bx	bin_window X coordinate	
by	bin_window Y coordinate	
wx	return location for widget X coordinate.	[out]
wy	return location for widget Y coordinate.	[out]

Since: 2.12

---

gtk_tree_view_convert_tree_to_bin_window_coords ()

```
void
gtk_tree_view_convert_tree_to_bin_window_coords
                                (GtkTreeView *tree_view,
                                 gint tx,
                                 gint ty,
                                 gint *bx,
                                 gint *by);
```

Converts tree coordinates (coordinates in full scrollable area of the tree) to bin_window coordinates.

Parameters

tree_view	a GtkTreeView	
tx	tree X coordinate	
ty	tree Y coordinate	
bx	return location for X coordinate relative to bin_window.	[out]
by	return location for Y coordinate	[out]

relative to bin_window.

Since: 2.12

---

gtk_tree_view_convert_tree_to_widget_coords ()

```
void
gtk_tree_view_convert_tree_to_widget_coords
                                (GtkTreeView *tree_view,
                                 gint tx,
                                 gint ty,
                                 gint *wx,
                                 gint *wy);
```

Converts tree coordinates (coordinates in full scrollable area of the tree) to widget coordinates.

Parameters

tree_view	a GtkTreeView	
tx	X coordinate relative to the tree	
ty	Y coordinate relative to the tree	
wx	return location for widget X coordinate.	[out]
wy	return location for widget Y coordinate.	[out]

Since: 2.12

---

gtk_tree_view_convert_widget_to_bin_window_coords ()

```
void
gtk_tree_view_convert_widget_to_bin_window_coords
                                (GtkTreeView *tree_view,
                                 gint wx,
                                 gint wy,
                                 gint *bx,
                                 gint *by);
```

Converts widget coordinates to coordinates for the bin_window (see `gtk_tree_view_get_bin_window()`).

Parameters

tree_view	a GtkTreeView	
wx	X coordinate relative to the widget	
wy	Y coordinate relative to the widget	
bx	return location for bin_window X coordinate.	[out]
by	return location for bin_window Y coordinate.	[out]

Since: 2.12

---

gtk_tree_view_convert_widget_to_tree_coords ()

```
void
gtk_tree_view_convert_widget_to_tree_coords
                                (GtkTreeView *tree_view,
                                 gint wx,
                                 gint wy,
                                 gint *tx,
                                 gint *ty);
```

Converts widget coordinates to coordinates for the tree (the full scrollable area of the tree).

Parameters

tree_view	a GtkTreeView	
wx	X coordinate relative to the widget	
wy	Y coordinate relative to the widget	
tx	return location for tree X coordinate.	[out]
ty	return location for tree Y coordinate.	[out]

Since: 2.12

---

gtk_tree_view_enable_model_drag_dest ()

```
void
gtk_tree_view_enable_model_drag_dest (GtkTreeView *tree_view,
                                      const GtkTargetEntry *targets,
                                      gint n_targets,
                                      GdkDragAction actions);
```

Turns `tree_view` into a drop destination for automatic DND. Calling this method sets "reorderable" to `FALSE`.

Parameters

tree_view	a GtkTreeView	
targets	the table of targets that the drag will support.	[array length=n_targets]
n_targets	the number of items in `targets`	
actions	the bitmask of possible actions for a drag from this widget	

---

gtk_tree_view_enable_model_drag_source ()

```
void
gtk_tree_view_enable_model_drag_source
                                (GtkTreeView *tree_view,
                                 GdkModifierType start_button_mask,
                                 const GtkTargetEntry *targets,
                                 gint n_targets,
                                 GdkDragAction actions);
```

Turns `tree_view` into a drag source for automatic DND. Calling this method sets "reorderable" to `FALSE`.

Parameters

tree_view	a GtkTreeView
start_button_mask	Mask of allowed buttons to start drag
targets	the table of targets that the drag will support. [array length=n_targets]
n_targets	the number of items in `targets`
actions	the bitmask of possible actions for a drag from this widget

---

gtk_tree_view_unset_rows_drag_source ()

```
void
gtk_tree_view_unset_rows_drag_source (GtkTreeView *tree_view);
```

Undoes the effect of `gtk_tree_view_enable_model_drag_source()`. Calling this method sets "reorderable" to `FALSE`.

Parameters

tree_view	a GtkTreeView

---

gtk_tree_view_unset_rows_drag_dest ()

```
void
gtk_tree_view_unset_rows_drag_dest (GtkTreeView *tree_view);
```

Undoes the effect of `gtk_tree_view_enable_model_drag_dest()`. Calling this method sets "reorderable" to `FALSE`.

Parameters

tree_view	a GtkTreeView

---

gtk_tree_view_set_drag_dest_row ()

```
void
gtk_tree_view_set_drag_dest_row (GtkTreeView *tree_view,
                                 GtkTreePath *path,
                                 GtkTreeViewDropPosition pos);
```

Sets the row that is highlighted for feedback. If `path` is `NULL`, an existing highlight is removed.

Parameters

tree_view	a GtkTreeView	
path	The path of the row to highlight, or `NULL`.	[allow-none]
pos	Specifies whether to drop before, after or into the row	

---

gtk_tree_view_get_drag_dest_row ()

```
void
gtk_tree_view_get_drag_dest_row (GtkTreeView *tree_view,
                                 GtkTreePath **path,
                                 GtkTreeViewDropPosition *pos);
```

Gets information about the row that is highlighted for feedback.

Parameters

tree_view	a GtkTreeView	
path	Return location for the path of the highlighted row, or `NULL`.	[out][optional][nullable]
pos	Return location for the drop position, or `NULL`.	[out][optional]

---

gtk_tree_view_get_dest_row_at_pos ()

```
gboolean
gtk_tree_view_get_dest_row_at_pos (GtkTreeView *tree_view,
                                   gint drag_x,
                                   gint drag_y,
                                   GtkTreePath **path,
                                   GtkTreeViewDropPosition *pos);
```

Determines the destination row for a given position. `drag_x` and `drag_y` are expected to be in widget coordinates. This function is only meaningful if `tree_view` is realized. Therefore this function will always return `FALSE` if `tree_view` is not realized or does not have a model.

Parameters

tree_view	a GtkTreeView	
drag_x	the position to determine the destination row for	
drag_y	the position to determine the destination row for	
path	Return location for the path of the highlighted row, or `NULL`.	[out][optional][nullable]
pos	Return location for the drop position, or `NULL`.	[out][optional]

Returns

whether there is a row at the given position, `TRUE` if this is indeed the case.

---

gtk_tree_view_create_row_drag_icon ()

```
cairo_surface_t *
gtk_tree_view_create_row_drag_icon (GtkTreeView *tree_view,
                                    GtkTreePath *path);
```

Creates a cairo_surface_t representation of the row at `path` . This image is used for a drag icon.

Parameters

tree_view	a GtkTreeView
path	a GtkTreePath in `tree_view`

Returns

a newly-allocated surface of the drag icon.

[transfer full]

---

gtk_tree_view_set_enable_search ()

```
void
gtk_tree_view_set_enable_search (GtkTreeView *tree_view,
                                 gboolean enable_search);
```

If `enable_search` is set, then the user can type in text to search through the tree interactively (this is sometimes called "typeahead find").

Note that even if this is `FALSE`, the user can still initiate a search using the “start-interactive-search” key binding.

Parameters

tree_view	A GtkTreeView
enable_search	`TRUE`, if the user can search interactively

---

gtk_tree_view_get_enable_search ()

```
gboolean
gtk_tree_view_get_enable_search (GtkTreeView *tree_view);
```

Returns whether or not the tree allows to start interactive searching by typing in text.

Parameters

tree_view	A GtkTreeView

Returns

whether or not to let the user search interactively

---

gtk_tree_view_get_search_column ()

```
gint
gtk_tree_view_get_search_column (GtkTreeView *tree_view);
```

Gets the column searched on by the interactive search code.

Parameters

tree_view	A GtkTreeView

Returns

the column the interactive search code searches in.

---

gtk_tree_view_set_search_column ()

```
void
gtk_tree_view_set_search_column (GtkTreeView *tree_view,
                                 gint column);
```

Sets `column` as the column where the interactive search code should search in for the current model.

If the search column is set, users can use the “start-interactive-search” key binding to bring up search popup. The enable-search property controls whether simply typing text will also start an interactive search.

Note that `column` refers to a column of the current model. The search column is reset to -1 when the model is changed.

Parameters

tree_view	A GtkTreeView
column	the column of the model to search in, or -1 to disable searching

---

gtk_tree_view_get_search_equal_func ()

```
GtkTreeViewSearchEqualFunc
gtk_tree_view_get_search_equal_func (GtkTreeView *tree_view);
```

Returns the compare function currently in use.

[skip]

Parameters

tree_view	A GtkTreeView	

Returns

the currently used compare function for the search code.

---

gtk_tree_view_set_search_equal_func ()

```
void
gtk_tree_view_set_search_equal_func (GtkTreeView *tree_view,
                                     GtkTreeViewSearchEqualFunc search_equal_func,
                                     gpointer search_user_data,
                                     GDestroyNotify search_destroy);
```

Sets the compare function for the interactive search capabilities; note that somewhat like `strcmp()` returning 0 for equality GtkTreeViewSearchEqualFunc returns `FALSE` on matches.

Parameters

tree_view	A GtkTreeView	
search_equal_func	the compare function to use during the search	
search_user_data	user data to pass to `search_equal_func` , or `NULL`.	[allow-none]
search_destroy	Destroy notifier for `search_user_data` , or `NULL`.	[allow-none]

---

gtk_tree_view_get_search_entry ()

```
GtkEntry *
gtk_tree_view_get_search_entry (GtkTreeView *tree_view);
```

Returns the GtkEntry which is currently in use as interactive search entry for `tree_view` . In case the built-in entry is being used, `NULL` will be returned.

Parameters

tree_view	A GtkTreeView	

Returns

the entry currently in use as search entry.

[transfer none]

Since: 2.10

---

gtk_tree_view_set_search_entry ()

```
void
gtk_tree_view_set_search_entry (GtkTreeView *tree_view,
                                GtkEntry *entry);
```

Sets the entry which the interactive search code will use for this `tree_view`. This is useful when you want to provide a search entry in our interface at all time at a fixed position. Passing `NULL` for `entry` will make the interactive search code use the built-in popup entry again.

Parameters

tree_view	A GtkTreeView
entry	the entry the interactive search code [allow-none] of `tree_view` should use or `NULL`.

Since: 2.10

GtkTreeViewSearchPositionFunc ()

```
void
(*GtkTreeViewSearchPositionFunc) (GtkTreeView *tree_view,
                                  GtkWidget *search_dialog,
                                  gpointer user_data);
```

---

gtk_tree_view_get_search_position_func ()

```
GtkTreeViewSearchPositionFunc
gtk_tree_view_get_search_position_func
                               (GtkTreeView *tree_view);
```

Returns the positioning function currently in use.

[skip]

Parameters

tree_view	A GtkTreeView

Returns

the currently used function for positioning the search dialog.

Since: 2.10

---

gtk_tree_view_set_search_position_func ()

```
void
gtk_tree_view_set_search_position_func
                                (GtkTreeView *tree_view,
                                 GtkTreeViewSearchPositionFunc func,
                                 gpointer data,
                                 GDestroyNotify destroy);
```

Sets the function to use when positioning the search dialog.

Parameters

tree_view	A GtkTreeView	
func	the function to use to position the search dialog, or `NULL` to use the default search position function.	[allow-none]
data	user data to pass to `func` , or `NULL`.	[allow-none]
destroy	Destroy notifier for `data` , or `NULL`.	[allow-none]

Since: 2.10

---

gtk_tree_view_get_fixed_height_mode ()

```
gboolean
gtk_tree_view_get_fixed_height_mode (GtkTreeView *tree_view);
```

Returns whether fixed height mode is turned on for `tree_view` .

Parameters

tree_view	a GtkTreeView

Returns

`TRUE` if `tree_view` is in fixed height mode

Since: 2.6

---

gtk_tree_view_set_fixed_height_mode ()

```
void
gtk_tree_view_set_fixed_height_mode (GtkTreeView *tree_view,
                                     gboolean enable);
```

Enables or disables the fixed height mode of `tree_view` . Fixed height mode speeds up GtkTreeView by assuming that all rows have the same height. Only enable this option if all rows are the same height and all columns are of type `GTK_TREE_VIEW_COLUMN_FIXED`.

Parameters

tree_view	a GtkTreeView
enable	TRUE to enable fixed height mode

Since: 2.6

---

gtk_tree_view_get_hover_selection ()

```
gboolean
gtk_tree_view_get_hover_selection (GtkTreeView *tree_view);
```

Returns whether hover selection mode is turned on for tree_view .

Parameters

tree_view	a GtkTreeView

Returns

TRUE if tree_view is in hover selection mode

Since: 2.6

---

gtk_tree_view_set_hover_selection ()

```
void
gtk_tree_view_set_hover_selection (GtkTreeView *tree_view,
                                   gboolean hover);
```

Enables or disables the hover selection mode of tree_view . Hover selection makes the selected row follow the pointer. Currently, this works only for the selection modes GTK_SELECTION_SINGLE and GTK_SELECTION_BROWSE.

Parameters

tree_view	a GtkTreeView
hover	TRUE to enable hover selection mode

Since: 2.6

---

gtk_tree_view_get_hover_expand ()

```
gboolean
gtk_tree_view_get_hover_expand (GtkTreeView *tree_view);
```

Returns whether hover expansion mode is turned on for tree_view .

Parameters

tree_view	a GtkTreeView

Returns

`TRUE` if `tree_view` is in hover expansion mode

Since: 2.6

---

gtk_tree_view_set_hover_expand ()

```
void
gtk_tree_view_set_hover_expand (GtkTreeView *tree_view,
                                gboolean expand);
```

Enables or disables the hover expansion mode of `tree_view` . Hover expansion makes rows expand or collapse if the pointer moves over them.

Parameters

tree_view	a GtkTreeView
expand	`TRUE` to enable hover selection mode

Since: 2.6

GtkTreeDestroyCountFunc ()

```
void
(*GtkTreeDestroyCountFunc) (GtkTreeView *tree_view,
                            GtkTreePath *path,
                            gint children,
                            gpointer user_data);
```

---

gtk_tree_view_set_destroy_count_func ()

```
void
gtk_tree_view_set_destroy_count_func (GtkTreeView *tree_view,
                                      GtkTreeDestroyCountFunc func,
                                      gpointer data,
                                      GDestroyNotify destroy);
```

`gtk_tree_view_set_destroy_count_func` has been deprecated since version 3.4 and should not be used in newly-written code.

Accessibility does not need the function anymore.

This function should almost never be used. It is meant for private use by ATK for determining the number of visible children that are removed when the user collapses a row, or a row is deleted.

Parameters

tree_view	A GtkTreeView	
func	Function to be called when a view row is destroyed, or `NULL`.	[allow-none]
data	User data to be passed to `func` , or `NULL`.	[allow-none]
destroy	Destroy notifier for `data` , or `NULL`.	[allow-none]

GtkTreeViewRowSeparatorFunc ()

```
gboolean
(*GtkTreeViewRowSeparatorFunc) (GtkTreeModel *model,
                                GtkTreeIter *iter,
                                gpointer data);
```

Function type for determining whether the row pointed to by `iter` should be rendered as a separator. A common way to implement this is to have a boolean column in the model, whose values the GtkTreeViewRowSeparatorFunc returns.

Parameters

model	the GtkTreeModel	
iter	a GtkTreeIter pointing at a row in `model`	
data	user data.	[closure]

Returns

`TRUE` if the row is a separator

---

gtk_tree_view_get_row_separator_func ()

```
GtkTreeViewRowSeparatorFunc
gtk_tree_view_get_row_separator_func (GtkTreeView *tree_view);
```

Returns the current row separator function.

[skip]

Parameters

tree_view	a GtkTreeView	

Returns

the current row separator function.

Since: 2.6

---

gtk_tree_view_set_row_separator_func ()

```
void
gtk_tree_view_set_row_separator_func (GtkTreeView *tree_view,
                                      GtkTreeViewRowSeparatorFunc func,
                                      gpointer data,
                                      GDestroyNotify destroy);
```

Sets the row separator function, which is used to determine whether a row should be drawn as a separator. If the row separator function is `NULL`, no separators are drawn. This is the default value.

Parameters

tree_view	a GtkTreeView	
func	a GtkTreeViewRowSeparatorFunc.	[allow-none]
data	user data to pass to `func` , or `NULL`.	[allow-none]
destroy	destroy notifier for `data` , or `NULL`.	[allow-none]

Since: 2.6

---

gtk_tree_view_get_rubber_banding ()

```
gboolean
gtk_tree_view_get_rubber_banding (GtkTreeView *tree_view);
```

Returns whether rubber banding is turned on for `tree_view` . If the selection mode is GTK_SELECTION_MULTIPLE, rubber banding will allow the user to select multiple rows by dragging the mouse.

Parameters

tree_view	a GtkTreeView

Returns

`TRUE` if rubber banding in `tree_view` is enabled.

Since: 2.10

---

gtk_tree_view_set_rubber_banding ()

```
void
gtk_tree_view_set_rubber_banding (GtkTreeView *tree_view,
                                  gboolean enable);
```

Enables or disables rubber banding in `tree_view` . If the selection mode is GTK_SELECTION_MULTIPLE, rubber banding will allow the user to select multiple rows by dragging the mouse.

Parameters

tree_view	a GtkTreeView
enable	TRUE to enable rubber banding

Since: 2.10

---

gtk_tree_view_is_rubber_banding_active ()

```
gboolean
gtk_tree_view_is_rubber_banding_active
                                (GtkTreeView *tree_view);
```

Returns whether a rubber banding operation is currently being done in tree_view .

Parameters

tree_view	a GtkTreeView

Returns

TRUE if a rubber banding operation is currently being done in tree_view .

Since: 2.12

---

gtk_tree_view_get_enable_tree_lines ()

```
gboolean
gtk_tree_view_get_enable_tree_lines (GtkTreeView *tree_view);
```

Returns whether or not tree lines are drawn in tree_view .

Parameters

tree_view	a GtkTreeView.

Returns

TRUE if tree lines are drawn in tree_view , FALSE otherwise.

Since: 2.10

---

gtk_tree_view_set_enable_tree_lines ()

```
void
gtk_tree_view_set_enable_tree_lines (GtkTreeView *tree_view,
                                     gboolean enabled);
```

Sets whether to draw lines interconnecting the expanders in tree_view . This does not have any visible effects for lists.

Parameters

tree_view	a GtkTreeView
enabled	TRUE to enable tree line drawing, FALSE otherwise.

Since: 2.10

---

gtk_tree_view_get_grid_lines ()

```
GtkTreeViewGridLines
gtk_tree_view_get_grid_lines (GtkTreeView *tree_view);
```

Returns which grid lines are enabled in tree_view .

Parameters

tree_view	a GtkTreeView

Returns

a GtkTreeViewGridLines value indicating which grid lines are enabled.

Since: 2.10

---

gtk_tree_view_set_grid_lines ()

```
void
gtk_tree_view_set_grid_lines (GtkTreeView *tree_view,
                              GtkTreeViewGridLines grid_lines);
```

Sets which grid lines to draw in tree_view .

Parameters

tree_view	a GtkTreeView
grid_lines	a GtkTreeViewGridLines value indicating which grid lines to enable.

Since: 2.10

---

gtk_tree_view_set_tooltip_row ()

```
void
gtk_tree_view_set_tooltip_row (GtkTreeView *tree_view,
                               GtkTooltip *tooltip,
                               GtkTreePath *path);
```

Sets the tip area of `tooltip` to be the area covered by the row at `path` . See also `gtk_tree_view_set_tooltip_column()` for a simpler alternative. See also `gtk_tooltip_set_tip_area()`.

Parameters

tree_view	a GtkTreeView	
tooltip	a GtkTooltip	
path	a GtkTreePath	

Since: 2.12

---

gtk_tree_view_set_tooltip_cell ()

```
void
gtk_tree_view_set_tooltip_cell (GtkTreeView *tree_view,
                                GtkTooltip *tooltip,
                                GtkTreePath *path,
                                GtkTreeViewColumn *column,
                                GtkCellRenderer *cell);
```

Sets the tip area of `tooltip` to the area `path` , `column` and `cell` have in common. For example if `path` is `NULL` and `column` is set, the tip area will be set to the full area covered by `column` . See also `gtk_tooltip_set_tip_area()`.

Note that if `path` is not specified and `cell` is set and part of a column containing the expander, the tooltip might not show and hide at the correct position. In such cases `path` must be set to the current node under the mouse cursor for this function to operate correctly.

See also `gtk_tree_view_set_tooltip_column()` for a simpler alternative.

Parameters

tree_view	a GtkTreeView	
tooltip	a GtkTooltip	
path	a GtkTreePath or `NULL`.	[allow-none]
column	a GtkTreeViewColumn or `NULL`.	[allow-none]
cell	a GtkCellRenderer or `NULL`.	[allow-none]

Since: 2.12

---

gtk_tree_view_get_tooltip_context ()

```
gboolean
gtk_tree_view_get_tooltip_context (GtkTreeView *tree_view,
                                   gint *x,
                                   gint *y,
                                   gboolean keyboard_tip,
                                   GtkTreeModel **model,
                                   GtkTreePath **path,
                                   GtkTreeIter *iter);
```

This function is supposed to be used in a “query-tooltip” signal handler for GtkTreeView. The `x` , `y` and `keyboard_tip` values which are received in the signal handler, should be passed to this function without

modification.

The return value indicates whether there is a tree view row at the given coordinates (`TRUE`) or not (`FALSE`) for mouse tooltips. For keyboard tooltips the row returned will be the cursor row. When `TRUE`, then any of `model`, `path` and `iter` which have been provided will be set to point to that row and the corresponding model. `x` and `y` will always be converted to be relative to `tree_view` 's bin_window if `keyboard_tooltip` is `FALSE`.

Parameters

tree_view	a GtkTreeView	
x	the x coordinate (relative to widget coordinates).	[inout]
y	the y coordinate (relative to widget coordinates).	[inout]
keyboard_tip	whether this is a keyboard tooltip or not	
model	a pointer to receive a GtkTreeModel or `NULL`.	[out][optional][nullable][transfer none]
path	a pointer to receive a GtkTreePath or `NULL`.	[out][optional]
iter	a pointer to receive a GtkTreeIter or `NULL`.	[out][optional]

Returns

whether or not the given tooltip context points to a row.

Since: 2.12

---

gtk_tree_view_get_tooltip_column ()

```
gint
gtk_tree_view_get_tooltip_column (GtkTreeView *tree_view);
```

Returns the column of `tree_view` 's model which is being used for displaying tooltips on `tree_view` 's rows.

Parameters

tree_view	a GtkTreeView

Returns

the index of the tooltip column that is currently being used, or -1 if this is disabled.

Since: 2.12

---

gtk_tree_view_set_tooltip_column ()

```
void
gtk_tree_view_set_tooltip_column (GtkTreeView *tree_view,
                                  gint column);
```

If you only plan to have simple (text-only) tooltips on full rows, you can use this function to have GtkTreeView handle these automatically for you. `column` should be set to the column in `tree_view` 's model containing the tooltip texts, or -1 to disable this feature.

When enabled, "has-tooltip" will be set to `TRUE` and `tree_view` will connect a "query-tooltip" signal handler.

Note that the signal handler sets the text with `gtk_tooltip_set_markup()`, so &, <, etc have to be escaped in the text.

Parameters

tree_view	a GtkTreeView
column	an integer, which is a valid column number for `tree_view` 's model

Since: 2.12

Types and Values

struct GtkTreeView

```
struct GtkTreeView;
```

---

enum GtkTreeViewDropPosition

An enum for determining where a dropped row goes.

Members

GTK_TREE_VIEW_DROP_BEFORE	dropped row is inserted before
GTK_TREE_VIEW_DROP_AFTER	dropped row is inserted after
GTK_TREE_VIEW_DROP_INTO_OR_BEFORE	dropped row becomes a child or is inserted before
GTK_TREE_VIEW_DROP_INTO_OR_AFTER	dropped row becomes a child or is inserted after

---

GtkTreeViewPrivate

```
typedef struct _GtkTreeViewPrivate GtkTreeViewPrivate;
```

enum GtkTreeViewGridLines

Used to indicate which grid lines to draw in a tree view.

Members

GTK_TREE_VIEW_GRID_LINES_NONE	No grid lines.
GTK_TREE_VIEW_GRID_LINES_HORIZONTAL	Horizontal grid lines.
GTK_TREE_VIEW_GRID_LINES_VERTICAL	Vertical grid lines.
GTK_TREE_VIEW_GRID_LINES_BOTH	Horizontal and vertical grid lines.

Property Details

The `“activate-on-single-click”` property

```
“activate-on-single-click” gboolean
```

The activate-on-single-click property specifies whether the "row-activated" signal will be emitted after a single click.

Flags: Read / Write

Default value: FALSE

Since: 3.8

The `“enable-grid-lines”` property

```
“enable-grid-lines”        GtkTreeViewGridLines
```

Whether grid lines should be drawn in the tree view.

Flags: Read / Write

Default value: GTK_TREE_VIEW_GRID_LINES_NONE

The `“enable-search”` property

```
“enable-search”            gboolean
```

View allows user to search through columns interactively.

Flags: Read / Write

Default value: TRUE

The “enable-tree-lines” property

“enable-tree-lines” gboolean

Whether tree lines should be drawn in the tree view.

Flags: Read / Write

Default value: FALSE

The “expander-column” property

“expander-column” GtkTreeViewColumn *

Set the column for the expander column.

Flags: Read / Write

The “fixed-height-mode” property

“fixed-height-mode” gboolean

Setting the ::fixed-height-mode property to TRUE speeds up GtkTreeView by assuming that all rows have the same height. Only enable this option if all rows are the same height. Please see gtk_tree_view_set_fixed_height_mode() for more information on this option.

Flags: Read / Write

Default value: FALSE

Since: 2.4

The “headers-clickable” property

“headers-clickable” gboolean

Column headers respond to click events.

Flags: Read / Write

Default value: TRUE

The “headers-visible” property

“headers-visible” gboolean

Show the column header buttons.

Flags: Read / Write

Default value: TRUE

The “hover-expand” property

“hover-expand” gboolean

Enables or disables the hover expansion mode of tree_view . Hover expansion makes rows expand or collapse if the pointer moves over them.

This mode is primarily intended for treeviews in popups, e.g. in GtkComboBox or GtkEntryCompletion.

Flags: Read / Write

Default value: FALSE

Since: 2.6

The “hover-selection” property

“hover-selection” gboolean

Enables or disables the hover selection mode of tree_view . Hover selection makes the selected row follow the pointer. Currently, this works only for the selection modes GTK_SELECTION_SINGLE and GTK_SELECTION_BROWSE.

This mode is primarily intended for treeviews in popups, e.g. in GtkComboBox or GtkEntryCompletion.

Flags: Read / Write

Default value: FALSE

Since: 2.6

The “level-indentation” property

“level-indentation” gint

Extra indentation for each level.

Flags: Read / Write

Allowed values: >= 0

Default value: 0

Since: 2.12

The “model” property

“model” GtkTreeModel *

The model for the tree view.

Flags: Read / Write

The “reorderable” property

“reorderable” gboolean

View is reorderable.

Flags: Read / Write

Default value: FALSE

The “rubber-banding” property

“rubber-banding” gboolean

Whether to enable selection of multiple items by dragging the mouse pointer.

Flags: Read / Write

Default value: FALSE

The “rules-hint” property

“rules-hint” gboolean

Sets a hint to the theme to draw rows in alternating colors.

GtkTreeView:rules-hint has been deprecated since version 3.14 and should not be used in newly-written code.

The theme is responsible for drawing rows using zebra striping

Flags: Read / Write

Default value: FALSE

The “search-column” property

“search-column” gint

Model column to search through during interactive search.

Flags: Read / Write

Allowed values: >= -1

Default value: -1

The “show-expanders” property

“show-expanders” gboolean

TRUE if the view has expanders.

Flags: Read / Write

Default value: TRUE

Since: 2.12

The “tooltip-column” property

“tooltip-column” gint

The column in the model containing the tooltip texts for the rows.

Flags: Read / Write

Allowed values: >= -1

Default value: -1

Style Property Details

The “allow-rules” style property

“allow-rules” gboolean

Allow drawing of alternating color rows.

Flags: Read

Default value: TRUE

The “even-row-color” style property

“even-row-color” GdkColor *

Color to use for even rows.

Flags: Read

The “expander-size” style property

“expander-size” gint

Size of the expander arrow.

Flags: Read

Allowed values: >= 0

Default value: 14

The “grid-line-pattern” style property

“grid-line-pattern” gchar *

Dash pattern used to draw the tree view grid lines.

Flags: Read

Default value: "\001\001"

The “grid-line-width” style property

“grid-line-width” gint

Width, in pixels, of the tree view grid lines.

Flags: Read

Allowed values: >= 0

Default value: 1

The “horizontal-separator” style property

“horizontal-separator” gint

Horizontal space between cells. Must be an even number.

Flags: Read

Allowed values: >= 0

Default value: 2

The “indent-expanders” style property

“indent-expanders” gboolean

Make the expanders indented.

Flags: Read

Default value: TRUE

The “odd-row-color” style property

“odd-row-color” GdkColor *

Color to use for odd rows.

Flags: Read

The “tree-line-pattern” style property

```
  “tree-line-pattern”        gchar *
```

Dash pattern used to draw the tree view lines.

Flags: Read

Default value: "\001\001"

The “tree-line-width” style property

```
  “tree-line-width”          gint
```

Width, in pixels, of the tree view lines.

Flags: Read

Allowed values: >= 0

Default value: 1

The “vertical-separator” style property

```
  “vertical-separator”       gint
```

Vertical space between cells. Must be an even number.

Flags: Read

Allowed values: >= 0

Default value: 2

Signal Details

The “columns-changed” signal

```
void
user_function (GtkTreeView *tree_view,
               gpointer     user_data)
```

The number of columns of the treeview has changed.

Parameters

tree_view	the object on which the signal is emitted

user_data	user data set when the signal handler was connected.

Flags: Run Last

The “cursor-changed” signal

```
void
user_function (GtkTreeView *tree_view,
               gpointer     user_data)
```

The position of the cursor (focused cell) has changed.

Parameters

tree_view	the object on which the signal is emitted
user_data	user data set when the signal handler was connected.

Flags: Run Last

The “expand-collapse-cursor-row” signal

```
gboolean
user_function (GtkTreeView *treeview,
               gboolean     arg1,
               gboolean     arg2,
               gboolean     arg3,
               gpointer     user_data)
```

Flags: Action

The “move-cursor” signal

```
gboolean
user_function (GtkTreeView    *tree_view,
               GtkMovementStep step,
               gint            direction,
               gpointer        user_data)
```

The “move-cursor” signal is a keybinding signal which gets emitted when the user presses one of the cursor keys.

Applications should not connect to it, but may emit it with `g_signal_emit_by_name()` if they need to control the cursor programmatically. In contrast to `gtk_tree_view_set_cursor()` and `gtk_tree_view_set_cursor_on_cell()` when moving horizontally “move-cursor” does not reset the current selection.

Parameters

tree_view	the object on which the signal is emitted.
step	the granularity of the move, as a GtkMovementStep. `GTK_MOVEMENT_LOGICAL_POSITIONS`, `GTK_MOVEMENT_VISUAL_POSITIONS`, `GTK_MOVEMENT_DISPLAY_LINES`, `GTK_MOVEMENT_PAGES` and `GTK_MOVEMENT_BUFFER_ENDS` are supported. `GTK_MOVEMENT_LOGICAL_POSITIONS` and `GTK_MOVEMENT_VISUAL_POSITIONS` are treated identically.
direction	the direction to move: +1 to move forwards; -1 to move backwards. The resulting movement is undefined for all other values.
user_data	user data set when the signal handler was connected.

Returns

`TRUE` if `step` is supported, `FALSE` otherwise.

Flags: Action

The `"row-activated"` signal

```
void
user_function (GtkTreeView       *tree_view,
               GtkTreePath       *path,
               GtkTreeViewColumn *column,
               gpointer           user_data)
```

The "row-activated" signal is emitted when the method `gtk_tree_view_row_activated()` is called, when the user double clicks a treeview row with the "activate-on-single-click" property set to `FALSE`, or when the user single clicks a row when the "activate-on-single-click" property set to `TRUE`. It is also emitted when a non-editable row is selected and one of the keys: Space, Shift+Space, Return or Enter is pressed.

For selection handling refer to the tree widget conceptual overview as well as GtkTreeSelection.

Parameters

tree_view	the object on which the signal is emitted
path	the GtkTreePath for the activated row
column	the GtkTreeViewColumn in which

	the activation occurred
user_data	user data set when the signal handler was connected.

Flags: Action

The “row-collapsed” signal

```
void
user_function (GtkTreeView *tree_view,
               GtkTreeIter *iter,
               GtkTreePath *path,
               gpointer     user_data)
```

The given row has been collapsed (child nodes are hidden).

Parameters

tree_view	the object on which the signal is emitted
iter	the tree iter of the collapsed row
path	a tree path that points to the row
user_data	user data set when the signal handler was connected.

Flags: Run Last

The “row-expanded” signal

```
void
user_function (GtkTreeView *tree_view,
               GtkTreeIter *iter,
               GtkTreePath *path,
               gpointer     user_data)
```

The given row has been expanded (child nodes are shown).

Parameters

tree_view	the object on which the signal is emitted
iter	the tree iter of the expanded row
path	a tree path that points to the row
user_data	user data set when the signal handler was connected.

Flags: Run Last

The “select-all” signal

```
gboolean
```

```
user_function (GtkTreeView *treeview,
               gpointer     user_data)
```

Flags: Action

The “select-cursor-parent” signal

```
gboolean
user_function (GtkTreeView *treeview,
               gpointer     user_data)
```

Flags: Action

The “select-cursor-row” signal

```
gboolean
user_function (GtkTreeView *treeview,
               gboolean     arg1,
               gpointer     user_data)
```

Flags: Action

The “start-interactive-search” signal

```
gboolean
user_function (GtkTreeView *treeview,
               gpointer     user_data)
```

Flags: Action

The “test-collapse-row” signal

```
gboolean
user_function (GtkTreeView *tree_view,
               GtkTreeIter *iter,
               GtkTreePath *path,
               gpointer     user_data)
```

The given row is about to be collapsed (hide its children nodes). Use this signal if you need to control the collapsibility of individual rows.

Parameters

tree_view	the object on which the signal is emitted
iter	the tree iter of the row to collapse
path	a tree path that points to the row
user_data	user data set when the signal handler was connected.

Returns

FALSE to allow collapsing, TRUE to reject

Flags: Run Last

The “test-expand-row” signal

```
gboolean
user_function (GtkTreeView *tree_view,
               GtkTreeIter *iter,
               GtkTreePath *path,
               gpointer     user_data)
```

The given row is about to be expanded (show its children nodes). Use this signal if you need to control the expandability of individual rows.

Parameters

tree_view	the object on which the signal is emitted
iter	the tree iter of the row to expand
path	a tree path that points to the row
user_data	user data set when the signal handler was connected.

Returns

FALSE to allow expansion, TRUE to reject

Flags: Run Last

The “toggle-cursor-row” signal

```
gboolean
user_function (GtkTreeView *treeview,
               gpointer     user_data)
```

Flags: Action

The “unselect-all” signal

```
gboolean
user_function (GtkTreeView *treeview,
               gpointer     user_data)
```

Flags: Action

See Also

GtkTreeViewColumn, GtkTreeSelection, GtkTreeModel, GtkTreeView drag-and-drop, GtkTreeSortable, GtkTreeModelSort, GtkListStore, GtkTreeStore, GtkCellRenderer, GtkCellEditable, GtkCellRendererPixbuf, GtkCellRendererText, GtkCellRendererToggle

---

GtkTreeView drag-and-drop

GtkTreeView drag-and-drop — Interfaces for drag-and-drop support in GtkTreeView

Functions

gboolean	gtk_tree_drag_source_drag_data_delete ()
gboolean	gtk_tree_drag_source_drag_data_get ()
gboolean	gtk_tree_drag_source_row_draggable ()
gboolean	gtk_tree_drag_dest_drag_data_received ()
gboolean	gtk_tree_drag_dest_row_drop_possible ()
gboolean	gtk_tree_set_row_drag_data ()
gboolean	gtk_tree_get_row_drag_data ()

Types and Values

	GtkTreeDragSource
struct	GtkTreeDragSourceIface
	GtkTreeDragDest
struct	GtkTreeDragDestIface

Object Hierarchy

```
GInterface
├── GtkTreeDragDest
└── GtkTreeDragSource
```

Known Implementations

GtkTreeDragSource is implemented by GtkListStore, GtkTreeModelFilter, GtkTreeModelSort and GtkTreeStore.

GtkTreeDragDest is implemented by GtkListStore and GtkTreeStore.

Includes

```
#include <gtk/gtk.h>
```

Description

GTK+ supports Drag-and-Drop in tree views with a high-level and a low-level API.

The low-level API consists of the GTK+ DND API, augmented by some treeview utility functions: `gtk_tree_view_set_drag_dest_row()`, `gtk_tree_view_get_drag_dest_row()`, `gtk_tree_view_get_dest_row_at_pos()`, `gtk_tree_view_create_row_drag_icon()`, `gtk_tree_set_row_drag_data()` and `gtk_tree_get_row_drag_data()`. This API leaves a lot of flexibility, but nothing is done automatically, and implementing advanced features like hover-to-open-rows or autoscrolling on top of this API is a lot of work.

On the other hand, if you write to the high-level API, then all the bookkeeping of rows is done for you, as well as things like hover-to-open and auto-scroll, but your models have to implement the GtkTreeDragSource and GtkTreeDragDest interfaces.

Functions

gtk_tree_drag_source_drag_data_delete ()

```
gboolean
gtk_tree_drag_source_drag_data_delete (GtkTreeDragSource *drag_source,
                                       GtkTreePath *path);
```

Asks the GtkTreeDragSource to delete the row at `path` , because it was moved somewhere else via drag-and-drop. Returns `FALSE` if the deletion fails because `path` no longer exists, or for some model-specific reason. Should robustly handle a `path` no longer found in the model!

Parameters

drag_source	a GtkTreeDragSource
path	row that was being dragged

Returns

`TRUE` if the row was successfully deleted

---

gtk_tree_drag_source_drag_data_get ()

```
gboolean
gtk_tree_drag_source_drag_data_get (GtkTreeDragSource *drag_source,
                                    GtkTreePath *path,
                                    GtkSelectionData *selection_data);
```

Asks the GtkTreeDragSource to fill in `selection_data` with a representation of the row at `path` . `selection_data->target` gives the required type of the data. Should robustly handle a `path` no longer found in the model!

Parameters

drag_source	a GtkTreeDragSource
path	row that was dragged
selection_data	a GtkSelectionData to fill with data from the dragged row

Returns

`TRUE` if data of the required type was provided

---

gtk_tree_drag_source_row_draggable ()

```
gboolean
gtk_tree_drag_source_row_draggable (GtkTreeDragSource *drag_source,
                                    GtkTreePath *path);
```

Asks the GtkTreeDragSource whether a particular row can be used as the source of a DND operation. If the source doesn't implement this interface, the row is assumed draggable.

Parameters

drag_source	a GtkTreeDragSource
path	row on which user is initiating a drag

Returns

`TRUE` if the row can be dragged

---

gtk_tree_drag_dest_drag_data_received ()

```
gboolean
gtk_tree_drag_dest_drag_data_received (GtkTreeDragDest *drag_dest,
                                       GtkTreePath *dest,
                                       GtkSelectionData *selection_data);
```

Asks the GtkTreeDragDest to insert a row before the path `dest` , deriving the contents of the row from `selection_data` . If `dest` is outside the tree so that inserting before it is impossible, `FALSE` will be returned. Also, `FALSE` may be returned if the new row is not created for some model-specific reason. Should robustly handle a `dest` no longer found in the model!

Parameters

drag_dest	a GtkTreeDragDest
dest	row to drop in front of
selection_data	data to drop

Returns

whether a new row was created before position `dest`

---

gtk_tree_drag_dest_row_drop_possible ()

```
gboolean
gtk_tree_drag_dest_row_drop_possible (GtkTreeDragDest *drag_dest,
                                      GtkTreePath *dest_path,
                                      GtkSelectionData *selection_data);
```

Determines whether a drop is possible before the given `dest_path` , at the same depth as `dest_path` . i.e., can we drop the data in `selection_data` at that location. `dest_path` does not have to exist; the return value will almost certainly be `FALSE` if the parent of `dest_path` doesn't exist, though.

Parameters

drag_dest	a GtkTreeDragDest
dest_path	destination row
selection_data	the data being dragged

Returns

`TRUE` if a drop is possible before `dest_path`

---

gtk_tree_set_row_drag_data ()

```
gboolean
gtk_tree_set_row_drag_data (GtkSelectionData *selection_data,
                            GtkTreeModel *tree_model,
                            GtkTreePath *path);
```

Sets selection data of target type `GTK_TREE_MODEL_ROW`. Normally used in a drag_data_get handler.

Parameters

selection_data	some GtkSelectionData
tree_model	a GtkTreeModel
path	a row in `tree_model`

Returns

`TRUE` if the GtkSelectionData had the proper target type to allow us to set a tree row

---

gtk_tree_get_row_drag_data ()

```
gboolean
```

```
gtk_tree_get_row_drag_data (GtkSelectionData *selection_data,
                            GtkTreeModel **tree_model,
                            GtkTreePath **path);
```

Obtains a `tree_model` and `path` from selection data of target type `GTK_TREE_MODEL_ROW`. Normally called from a drag_data_received handler. This function can only be used if `selection_data` originates from the same process that's calling this function, because a pointer to the tree model is being passed around. If you aren't in the same process, then you'll get memory corruption. In the GtkTreeDragDest drag_data_received handler, you can assume that selection data of type `GTK_TREE_MODEL_ROW` is in from the current process. The returned path must be freed with `gtk_tree_path_free()`.

Parameters

selection_data	a GtkSelectionData	
tree_model	a GtkTreeModel.	[nullable][optional][transfer none][out]
path	row in `tree_model`.	[nullable][optional][out]

Returns

`TRUE` if `selection_data` had target type `GTK_TREE_MODEL_ROW` and is otherwise valid

Types and Values

GtkTreeDragSource

```
typedef struct _GtkTreeDragSource GtkTreeDragSource;
```

---

struct GtkTreeDragSourceIface

```
struct GtkTreeDragSourceIface {
  /* VTable - not signals */

  gboolean     (* row_draggable)        (GtkTreeDragSource   *drag_source,
                                         GtkTreePath         *path);

  gboolean     (* drag_data_get)        (GtkTreeDragSource   *drag_source,
                                         GtkTreePath         *path,
                                         GtkSelectionData    *selection_data);

  gboolean     (* drag_data_delete)     (GtkTreeDragSource *drag_source,
                                         GtkTreePath       *path);
};
```

Members

`row_draggable` ()	Asks the GtkTreeDragSource whether a particular row can be used as the source of a DND

	operation.
drag_data_get ()	Asks the GtkTreeDragSource to fill in selection_data with a representation of the row at path.
drag_data_delete ()	Asks the GtkTreeDragSource to delete the row at path, because it was moved somewhere else via drag-and-drop.

GtkTreeDragDest

```
typedef struct _GtkTreeDragDest GtkTreeDragDest;
```

---

struct GtkTreeDragDestIface

```
struct GtkTreeDragDestIface {
  /* VTable - not signals */

  gboolean     (* drag_data_received) (GtkTreeDragDest   *drag_dest,
                                       GtkTreePath       *dest,
                                       GtkSelectionData  *selection_data);

  gboolean     (* row_drop_possible)  (GtkTreeDragDest   *drag_dest,
                                       GtkTreePath       *dest_path,
                                  GtkSelectionData  *selection_data);
};
```

Members

drag_data_received ()	Asks the GtkTreeDragDest to insert a row before the path dest, deriving the contents of the row from selection_data.
row_drop_possible ()	Determines whether a drop is possible before the given dest_path, at the same depth as dest_path.

GtkCellView

GtkCellView — A widget displaying a single row of a GtkTreeModel

Functions

GtkWidget *	gtk_cell_view_new ()
GtkWidget *	gtk_cell_view_new_with_context ()
GtkWidget *	gtk_cell_view_new_with_text ()

GtkWidget *	gtk_cell_view_new_with_markup ()
GtkWidget *	gtk_cell_view_new_with_pixbuf ()
void	gtk_cell_view_set_model ()
GtkTreeModel *	gtk_cell_view_get_model ()
void	gtk_cell_view_set_displayed_row ()
GtkTreePath *	gtk_cell_view_get_displayed_row ()
gboolean	gtk_cell_view_get_size_of_row ()
void	gtk_cell_view_set_background_color ()
void	gtk_cell_view_set_background_rgba ()
void	gtk_cell_view_set_draw_sensitive ()
gboolean	gtk_cell_view_get_draw_sensitive ()
void	gtk_cell_view_set_fit_model ()
gboolean	gtk_cell_view_get_fit_model ()

Properties

gchar *	background	Write
GdkColor *	background-gdk	Read / Write
GdkRGBA *	background-rgba	Read / Write
gboolean	background-set	Read / Write
GtkCellArea *	cell-area	Read / Write / Construct Only
GtkCellAreaContext *	cell-area-context	Read / Write / Construct Only
gboolean	draw-sensitive	Read / Write
gboolean	fit-model	Read / Write
GtkTreeModel *	model	Read / Write

Types and Values

struct	GtkCellView
struct	GtkCellViewClass

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkCellView
```

Implemented Interfaces

GtkCellView implements AtkImplementorIface, GtkBuildable, GtkCellLayout and GtkOrientable.

Includes

```
#include <gtk/gtk.h>
```

Description

A GtkCellView displays a single row of a GtkTreeModel using a GtkCellArea and GtkCellAreaContext. A

GtkCellAreaContext can be provided to the GtkCellView at construction time in order to keep the cellview in context of a group of cell views, this ensures that the renderers displayed will be properly aligned with eachother (like the aligned cells in the menus of GtkComboBox).

GtkCellView is GtkOrientable in order to decide in which orientation the underlying GtkCellAreaContext should be allocated. Taking the GtkComboBox menu as an example, cellviews should be oriented horizontally if the menus are listed top-to-bottom and thus all share the same width but may have separate individual heights (left-to-right menus should be allocated vertically since they all share the same height but may have variable widths).

CSS nodes

GtkCellView has a single CSS node with name cellview.

Functions

gtk_cell_view_new ()

```
GtkWidget *
gtk_cell_view_new (void);
```

Creates a new GtkCellView widget.

Returns

A newly created GtkCellView widget.

Since: 2.6

---

gtk_cell_view_new_with_context ()

```
GtkWidget *
gtk_cell_view_new_with_context (GtkCellArea *area,
                                GtkCellAreaContext *context);
```

Creates a new GtkCellView widget with a specific GtkCellArea to layout cells and a specific GtkCellAreaContext.

Specifying the same context for a handfull of cells lets the underlying area synchronize the geometry for those cells, in this way alignments with cellviews for other rows are possible.

Parameters

area	the GtkCellArea to layout cells
context	the GtkCellAreaContext in which to calculate cell geometry

Returns

A newly created GtkCellView widget.

Since: 2.6

---

gtk_cell_view_new_with_text ()

```
GtkWidget *
gtk_cell_view_new_with_text (const gchar *text);
```

Creates a new GtkCellView widget, adds a GtkCellRendererText to it, and makes it show `text` .

Parameters

text	the text to display in the cell view

Returns

A newly created GtkCellView widget.

Since: 2.6

---

gtk_cell_view_new_with_markup ()

```
GtkWidget *
gtk_cell_view_new_with_markup (const gchar *markup);
```

Creates a new GtkCellView widget, adds a GtkCellRendererText to it, and makes it show `markup` . The text can be marked up with the Pango text markup language.

Parameters

markup	the text to display in the cell view

Returns

A newly created GtkCellView widget.

Since: 2.6

---

gtk_cell_view_new_with_pixbuf ()

```
GtkWidget *
gtk_cell_view_new_with_pixbuf (GdkPixbuf *pixbuf);
```

Creates a new GtkCellView widget, adds a GtkCellRendererPixbuf to it, and makes it show `pixbuf` .

Parameters

pixbuf	the image to display in the cell view

Returns

A newly created GtkCellView widget.

Since: 2.6

---

gtk_cell_view_set_model ()

```
void
gtk_cell_view_set_model (GtkCellView *cell_view,
                         GtkTreeModel *model);
```

Sets the model for `cell_view` . If `cell_view` already has a model set, it will remove it before setting the new model. If `model` is `NULL`, then it will unset the old model.

Parameters

cell_view	a GtkCellView	
model	a GtkTreeModel.	[allow-none]

Since: 2.6

---

gtk_cell_view_get_model ()

```
GtkTreeModel *
gtk_cell_view_get_model (GtkCellView *cell_view);
```

Returns the model for `cell_view` . If no model is used `NULL` is returned.

Parameters

cell_view	a GtkCellView

Returns

a GtkTreeModel used or `NULL`.

[nullable][transfer none]

Since: 2.16

---

gtk_cell_view_set_displayed_row ()

```
void
gtk_cell_view_set_displayed_row (GtkCellView *cell_view,
```

```
                                GtkTreePath *path);
```

Sets the row of the model that is currently displayed by the GtkCellView. If the path is unset, then the contents of the cellview “stick” at their last value; this is not normally a desired result, but may be a needed intermediate state if say, the model for the GtkCellView becomes temporarily empty.

Parameters

cell_view	a GtkCellView	
path	a GtkTreePath or `NULL` to unset.	[allow-none]

Since: 2.6

---

gtk_cell_view_get_displayed_row ()

```
GtkTreePath *
gtk_cell_view_get_displayed_row (GtkCellView *cell_view);
```

Returns a GtkTreePath referring to the currently displayed row. If no row is currently displayed, `NULL` is returned.

Parameters

cell_view	a GtkCellView

Returns

the currently displayed row or `NULL`.

[nullable][transfer full]

Since: 2.6

---

gtk_cell_view_get_size_of_row ()

```
gboolean
gtk_cell_view_get_size_of_row (GtkCellView *cell_view,
                               GtkTreePath *path,
                               GtkRequisition *requisition);
```

`gtk_cell_view_get_size_of_row` has been deprecated since version 3.0 and should not be used in newly-written code.

Combo box formerly used this to calculate the sizes for cellviews, now you can achieve this by either using the “fit-model” property or by setting the currently displayed row of the GtkCellView and using `gtk_widget_get_preferred_size()`.

Sets `requisition` to the size needed by `cell_view` to display the model row pointed to by `path` .

Parameters

cell_view	a GtkCellView	
path	a GtkTreePath	
requisition	return location for the size.	[out]

Returns

`TRUE`

Since: 2.6

---

gtk_cell_view_set_background_color ()

```
void
gtk_cell_view_set_background_color (GtkCellView *cell_view,
                                    const GdkColor *color);
```

`gtk_cell_view_set_background_color` has been deprecated since version 3.4 and should not be used in newly-written code.

Use `gtk_cell_view_set_background_rgba()` instead.

Sets the background color of `view` .

Parameters

cell_view	a GtkCellView
color	the new background color

Since: 2.6

---

gtk_cell_view_set_background_rgba ()

```
void
gtk_cell_view_set_background_rgba (GtkCellView *cell_view,
                                   const GdkRGBA *rgba);
```

Sets the background color of `cell_view` .

Parameters

cell_view	a GtkCellView
rgba	the new background color

Since: 3.0

---

gtk_cell_view_set_draw_sensitive ()

```
void
```

```
gtk_cell_view_set_draw_sensitive (GtkCellView *cell_view,
                                  gboolean draw_sensitive);
```

Sets whether `cell_view` should draw all of its cells in a sensitive state, this is used by GtkComboBox menus to ensure that rows with insensitive cells that contain children appear sensitive in the parent menu item.

Parameters

cell_view	a GtkCellView
draw_sensitive	whether to draw all cells in a sensitive state.

Since: 3.0

---

gtk_cell_view_get_draw_sensitive ()

```
gboolean
gtk_cell_view_get_draw_sensitive (GtkCellView *cell_view);
```

Gets whether `cell_view` is configured to draw all of its cells in a sensitive state.

Parameters

cell_view	a GtkCellView

Returns

whether `cell_view` draws all of its cells in a sensitive state

Since: 3.0

---

gtk_cell_view_set_fit_model ()

```
void
gtk_cell_view_set_fit_model (GtkCellView *cell_view,
                             gboolean fit_model);
```

Sets whether `cell_view` should request space to fit the entire GtkTreeModel.

This is used by GtkComboBox to ensure that the cell view displayed on the combo box's button always gets enough space and does not resize when selection changes.

Parameters

cell_view	a GtkCellView
fit_model	whether `cell_view` should request space for the whole model.

Since: 3.0

---

gtk_cell_view_get_fit_model ()

```
gboolean
gtk_cell_view_get_fit_model (GtkCellView *cell_view);
```

Gets whether `cell_view` is configured to request space to fit the entire GtkTreeModel.

Parameters

cell_view	a GtkCellView

Returns

whether `cell_view` requests space to fit the entire GtkTreeModel.

Since: 3.0

Types and Values

struct GtkCellView

```
struct GtkCellView;
```

---

struct GtkCellViewClass

```
struct GtkCellViewClass {
  GtkWidgetClass parent_class;
};
```

Members

Property Details

The “background” property

“background”	gchar *

Background color as a string.

Flags: Write

Default value: NULL

The “background-gdk” property

“background-gdk” GdkColor *

The background color as a GdkColor

GtkCellView:background-gdk has been deprecated since version 3.4 and should not be used in newly-written code.

Use “background-rgba” instead.

Flags: Read / Write

The “background-rgba” property

“background-rgba” GdkRGBA *

The background color as a GdkRGBA

Flags: Read / Write

Since: 3.0

The “background-set” property

“background-set” gboolean

Whether this tag affects the background color.

Flags: Read / Write

Default value: FALSE

The “cell-area” property

“cell-area” GtkCellArea *

The GtkCellArea rendering cells

If no area is specified when creating the cell view with gtk_cell_view_new_with_context() a horizontally oriented GtkCellAreaBox will be used.

since 3.0

Flags: Read / Write / Construct Only

The “cell-area-context” property

“cell-area-context” GtkCellAreaContext *

The GtkCellAreaContext used to compute the geometry of the cell view.

A group of cell views can be assigned the same context in order to ensure the sizes and cell alignments match across all the views with the same context.

GtkComboBox menus uses this to assign the same context to all cell views in the menu items for a single menu (each submenu creates its own context since the size of each submenu does not depend on parent or sibling menus).

since 3.0

Flags: Read / Write / Construct Only

The “draw-sensitive” property

“draw-sensitive” gboolean

Whether all cells should be draw as sensitive for this view regardless of the actual cell properties (used to make menus with submenus appear sensitive when the items in submenus might be insensitive).

since 3.0

Flags: Read / Write

Default value: FALSE

The “fit-model” property

“fit-model” gboolean

Whether the view should request enough space to always fit the size of every row in the model (used by the combo box to ensure the combo box size doesnt change when different items are selected).

since 3.0

Flags: Read / Write

Default value: FALSE

The “model” property

“model” GtkTreeModel *

The model for cell view

since 2.10

Flags: Read / Write

GtkIconView

GtkIconView — A widget which displays a list of icons in a grid

Functions

void	(*GtkIconViewForeachFunc) ()
GtkWidget *	gtk_icon_view_new ()
GtkWidget *	gtk_icon_view_new_with_area ()
GtkWidget *	gtk_icon_view_new_with_model ()
void	gtk_icon_view_set_model ()
GtkTreeModel *	gtk_icon_view_get_model ()
void	gtk_icon_view_set_text_column ()
gint	gtk_icon_view_get_text_column ()
void	gtk_icon_view_set_markup_column ()
gint	gtk_icon_view_get_markup_column ()
void	gtk_icon_view_set_pixbuf_column ()
gint	gtk_icon_view_get_pixbuf_column ()
GtkTreePath *	gtk_icon_view_get_path_at_pos ()
gboolean	gtk_icon_view_get_item_at_pos ()
void	gtk_icon_view_convert_widget_to_bin_window_coords ()
void	gtk_icon_view_set_cursor ()
gboolean	gtk_icon_view_get_cursor ()
void	gtk_icon_view_selected_foreach ()
void	gtk_icon_view_set_selection_mode ()
GtkSelectionMode	gtk_icon_view_get_selection_mode ()
void	gtk_icon_view_set_item_orientation ()
GtkOrientation	gtk_icon_view_get_item_orientation ()
void	gtk_icon_view_set_columns ()
gint	gtk_icon_view_get_columns ()
void	gtk_icon_view_set_item_width ()
gint	gtk_icon_view_get_item_width ()
void	gtk_icon_view_set_spacing ()
gint	gtk_icon_view_get_spacing ()
void	gtk_icon_view_set_row_spacing ()
gint	gtk_icon_view_get_row_spacing ()
void	gtk_icon_view_set_column_spacing ()
gint	gtk_icon_view_get_column_spacing ()
void	gtk_icon_view_set_margin ()
gint	gtk_icon_view_get_margin ()

void	gtk_icon_view_set_item_padding ()
gint	gtk_icon_view_get_item_padding ()
void	gtk_icon_view_set_activate_on_single_click ()
gboolean	gtk_icon_view_get_activate_on_single_click ()
gboolean	gtk_icon_view_get_cell_rect ()
void	gtk_icon_view_select_path ()
void	gtk_icon_view_unselect_path ()
gboolean	gtk_icon_view_path_is_selected ()
GList *	gtk_icon_view_get_selected_items ()
void	gtk_icon_view_select_all ()
void	gtk_icon_view_unselect_all ()
void	gtk_icon_view_item_activated ()
void	gtk_icon_view_scroll_to_path ()
gboolean	gtk_icon_view_get_visible_range ()
void	gtk_icon_view_set_tooltip_item ()
void	gtk_icon_view_set_tooltip_cell ()
gboolean	gtk_icon_view_get_tooltip_context ()
void	gtk_icon_view_set_tooltip_column ()
gint	gtk_icon_view_get_tooltip_column ()
gint	gtk_icon_view_get_item_row ()
gint	gtk_icon_view_get_item_column ()
void	gtk_icon_view_enable_model_drag_source ()
void	gtk_icon_view_enable_model_drag_dest ()
void	gtk_icon_view_unset_model_drag_source ()
void	gtk_icon_view_unset_model_drag_dest ()
void	gtk_icon_view_set_reorderable ()
gboolean	gtk_icon_view_get_reorderable ()
void	gtk_icon_view_set_drag_dest_item ()
void	gtk_icon_view_get_drag_dest_item ()
gboolean	gtk_icon_view_get_dest_item_at_pos ()
cairo_surface_t *	gtk_icon_view_create_drag_icon ()

Properties

gboolean	activate-on-single-click	Read / Write
GtkCellArea *	cell-area	Read / Write / Construct Only
gint	column-spacing	Read / Write
gint	columns	Read / Write
GtkOrientation	item-orientation	Read / Write
gint	item-padding	Read / Write
gint	item-width	Read / Write
gint	margin	Read / Write
gint	markup-column	Read / Write
GtkTreeModel *	model	Read / Write
gint	pixbuf-column	Read / Write
gboolean	reorderable	Read / Write
gint	row-spacing	Read / Write
GtkSelectionMode	selection-mode	Read / Write
gint	spacing	Read / Write
gint	text-column	Read / Write
gint	tooltip-column	Read / Write

Style Properties

guchar	selection-box-alpha	Read
GdkColor *	selection-box-color	Read

Signals

gboolean	activate-cursor-item	Action
void	item-activated	Run Last
gboolean	move-cursor	Action
void	select-all	Action
void	select-cursor-item	Action
void	selection-changed	Run First
void	toggle-cursor-item	Action
void	unselect-all	Action

Types and Values

struct	GtkIconView
enum	GtkIconViewDropPosition

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkContainer
            └── GtkIconView
```

Implemented Interfaces

GtkIconView implements AtkImplementorIface, GtkBuildable, GtkCellLayout and GtkScrollable.

Includes

```
#include <gtk/gtk.h>
```

Description

GtkIconView provides an alternative view on a GtkTreeModel. It displays the model as a grid of icons with labels. Like GtkTreeView, it allows to select one or multiple items (depending on the selection mode, see `gtk_icon_view_set_selection_mode()`). In addition to selection with the arrow keys, GtkIconView supports rubberband selection, which is controlled by dragging the pointer.

Note that if the tree model is backed by an actual tree store (as opposed to a flat list where the mapping to icons is obvious), GtkIconView will only display the first level of the tree and ignore the tree's branches.

CSS nodes

```
1                                                  iconview.view
2                                                  └── [rubberband]
```

GtkIconView has a single CSS node with name iconview and style class .view. For rubberband selection, a subnode with name rubberband is used.

Functions

GtkIconViewForeachFunc ()

```
void
(*GtkIconViewForeachFunc) (GtkIconView *icon_view,
                           GtkTreePath *path,
                           gpointer data);
```

A function used by `gtk_icon_view_selected_foreach()` to map all selected rows. It will be called on every selected row in the view.

Parameters

icon_view	a GtkIconView	
path	The GtkTreePath of a selected row	
data	user data.	[closure]

---

gtk_icon_view_new ()

```
GtkWidget *
gtk_icon_view_new (void);
```

Creates a new GtkIconView widget

Returns

A newly created GtkIconView widget

Since: 2.6

---

gtk_icon_view_new_with_area ()

```
GtkWidget *
gtk_icon_view_new_with_area (GtkCellArea *area);
```

Creates a new GtkIconView widget using the specified `area` to layout cells inside the icons.

Parameters

area	the GtkCellArea to use to layout

cells

Returns

A newly created GtkIconView widget

Since: 3.0

---

gtk_icon_view_new_with_model ()

```
GtkWidget *
gtk_icon_view_new_with_model (GtkTreeModel *model);
```

Creates a new GtkIconView widget with the model `model` .

Parameters

model	The model.	

Returns

A newly created GtkIconView widget.

Since: 2.6

---

gtk_icon_view_set_model ()

```
void
gtk_icon_view_set_model (GtkIconView *icon_view,
                         GtkTreeModel *model);
```

Sets the model for a GtkIconView. If the `icon_view` already has a model set, it will remove it before setting the new model. If `model` is `NULL`, then it will unset the old model.

Parameters

icon_view	A GtkIconView.	
model	The model.	[allow-none]

Since: 2.6

---

gtk_icon_view_get_model ()

```
GtkTreeModel *
gtk_icon_view_get_model (GtkIconView *icon_view);
```

Returns the model the GtkIconView is based on. Returns `NULL` if the model is unset.

Parameters

icon_view	a GtkIconView

Returns

A GtkTreeModel, or `NULL` if none is currently being used.

[nullable][transfer none]

Since: 2.6

---

gtk_icon_view_set_text_column ()

```
void
gtk_icon_view_set_text_column (GtkIconView *icon_view,
                               gint column);
```

Sets the column with text for `icon_view` to be `column` . The text column must be of type G_TYPE_STRING.

Parameters

icon_view	A GtkIconView.
column	A column in the currently used model, or -1 to display no text

Since: 2.6

---

gtk_icon_view_get_text_column ()

```
gint
gtk_icon_view_get_text_column (GtkIconView *icon_view);
```

Returns the column with text for `icon_view` .

Parameters

icon_view	A GtkIconView.

Returns

the text column, or -1 if it’s unset.

Since: 2.6

---

gtk_icon_view_set_markup_column ()

```
void
gtk_icon_view_set_markup_column (GtkIconView *icon_view,
```

```
                                gint column);
```

Sets the column with markup information for `icon_view` to be `column` . The markup column must be of type G_TYPE_STRING. If the markup column is set to something, it overrides the text column set by `gtk_icon_view_set_text_column()`.

Parameters

icon_view	A GtkIconView.
column	A column in the currently used model, or -1 to display no text

Since: 2.6

---

gtk_icon_view_get_markup_column ()

```
gint
gtk_icon_view_get_markup_column (GtkIconView *icon_view);
```

Returns the column with markup text for `icon_view` .

Parameters

icon_view	A GtkIconView.

Returns

the markup column, or -1 if it’s unset.

Since: 2.6

---

gtk_icon_view_set_pixbuf_column ()

```
void
gtk_icon_view_set_pixbuf_column (GtkIconView *icon_view,
                                 gint column);
```

Sets the column with pixbufs for `icon_view` to be `column` . The pixbuf column must be of type GDK_TYPE_PIXBUF

Parameters

icon_view	A GtkIconView.
column	A column in the currently used model, or -1 to disable

Since: 2.6

---

gtk_icon_view_get_pixbuf_column ()

```
gint
gtk_icon_view_get_pixbuf_column (GtkIconView *icon_view);
```

Returns the column with pixbufs for `icon_view`.

Parameters

icon_view	A GtkIconView.

Returns

the pixbuf column, or -1 if it's unset.

Since: 2.6

---

gtk_icon_view_get_path_at_pos ()

```
GtkTreePath *
gtk_icon_view_get_path_at_pos (GtkIconView *icon_view,
                               gint x,
                               gint y);
```

Finds the path at the point (`x`, `y`), relative to bin_window coordinates. See `gtk_icon_view_get_item_at_pos()`, if you are also interested in the cell at the specified position. See `gtk_icon_view_convert_widget_to_bin_window_coords()` for converting widget coordinates to bin_window coordinates.

Parameters

icon_view	A GtkIconView.
x	The x position to be identified
y	The y position to be identified

Returns

The GtkTreePath corresponding to the icon or `NULL` if no icon exists at that position.

[nullable][transfer full]

Since: 2.6

---

gtk_icon_view_get_item_at_pos ()

```
gboolean
gtk_icon_view_get_item_at_pos (GtkIconView *icon_view,
                               gint x,
                               gint y,
                               GtkTreePath **path,
                               GtkCellRenderer **cell);
```

Finds the path at the point (`x` , `y`), relative to bin_window coordinates. In contrast to `gtk_icon_view_get_path_at_pos()`, this function also obtains the cell at the specified position. The returned path should be freed with `gtk_tree_path_free()`. See `gtk_icon_view_convert_widget_to_bin_window_coords()` for converting widget coordinates to bin_window coordinates.

Parameters

icon_view	A GtkIconView.	
x	The x position to be identified	
y	The y position to be identified	
path	Return location for the path, or `NULL`.	[out][allow-none]
cell	Return location for the renderer responsible for the cell at (`x` , `y`), or `NULL`.	[out][allow-none][transfer none]

Returns

`TRUE` if an item exists at the specified position

Since: 2.8

---

gtk_icon_view_convert_widget_to_bin_window_coords ()

```
void
gtk_icon_view_convert_widget_to_bin_window_coords
                                (GtkIconView *icon_view,
                                 gint wx,
                                 gint wy,
                                 gint *bx,
                                 gint *by);
```

Converts widget coordinates to coordinates for the bin_window, as expected by e.g. `gtk_icon_view_get_path_at_pos()`.

Parameters

icon_view	a GtkIconView	
wx	X coordinate relative to the widget	
wy	Y coordinate relative to the widget	
bx	return location for bin_window X coordinate.	[out]
by	return location for bin_window Y coordinate.	[out]

Since: 2.12

---

gtk_icon_view_set_cursor ()

```
void
gtk_icon_view_set_cursor (GtkIconView *icon_view,
                          GtkTreePath *path,
                          GtkCellRenderer *cell,
                          gboolean start_editing);
```

Sets the current keyboard focus to be at `path` , and selects it. This is useful when you want to focus the user's attention on a particular item. If `cell` is not `NULL`, then focus is given to the cell specified by it. Additionally, if `start_editing` is `TRUE`, then editing should be started in the specified cell.

This function is often followed by `gtk_widget_grab_focus (icon_view)` in order to give keyboard focus to the widget. Please note that editing can only happen when the widget is realized.

Parameters

icon_view	A GtkIconView	
path	A GtkTreePath	
cell	One of the cell renderers of `icon_view` , or `NULL`.	[allow-none]
start_editing	`TRUE` if the specified cell should start being edited.	

Since: 2.8

---

gtk_icon_view_get_cursor ()

```
gboolean
gtk_icon_view_get_cursor (GtkIconView *icon_view,
                          GtkTreePath **path,
                          GtkCellRenderer **cell);
```

Fills in `path` and `cell` with the current cursor path and cell. If the cursor isn't currently set, then *`path` will be `NULL`. If no cell currently has focus, then *`cell` will be `NULL`.

The returned GtkTreePath must be freed with `gtk_tree_path_free()`.

Parameters

icon_view	A GtkIconView	
path	Return location for the current cursor path, or `NULL`.	[out][allow-none][transfer full]
cell	Return location the current focus cell, or `NULL`.	[out][allow-none][transfer none]

Returns

`TRUE` if the cursor is set.

Since: 2.8

---

gtk_icon_view_selected_foreach ()

```
void
gtk_icon_view_selected_foreach (GtkIconView *icon_view,
                                GtkIconViewForeachFunc func,
                                gpointer data);
```

Calls a function for each selected icon. Note that the model or selection cannot be modified from within this function.

Parameters

icon_view	A GtkIconView.	
func	The function to call for each selected icon.	[scope call]
data	User data to pass to the function.	

Since: 2.6

---

gtk_icon_view_set_selection_mode ()

```
void
gtk_icon_view_set_selection_mode (GtkIconView *icon_view,
                                  GtkSelectionMode mode);
```

Sets the selection mode of the `icon_view`.

Parameters

icon_view	A GtkIconView.
mode	The selection mode

Since: 2.6

---

gtk_icon_view_get_selection_mode ()

```
GtkSelectionMode
gtk_icon_view_get_selection_mode (GtkIconView *icon_view);
```

Gets the selection mode of the `icon_view`.

Parameters

icon_view	A GtkIconView.

Returns

the current selection mode

Since: 2.6

---

gtk_icon_view_set_item_orientation ()

```
void
gtk_icon_view_set_item_orientation (GtkIconView *icon_view,
                                    GtkOrientation orientation);
```

Sets the ::item-orientation property which determines whether the labels are drawn beside the icons instead of below.

Parameters

icon_view	a GtkIconView
orientation	the relative position of texts and icons

Since: 2.6

---

gtk_icon_view_get_item_orientation ()

```
GtkOrientation
gtk_icon_view_get_item_orientation (GtkIconView *icon_view);
```

Returns the value of the ::item-orientation property which determines whether the labels are drawn beside the icons instead of below.

Parameters

icon_view	a GtkIconView

Returns

the relative position of texts and icons

Since: 2.6

---

gtk_icon_view_set_columns ()

```
void
gtk_icon_view_set_columns (GtkIconView *icon_view,
                           gint columns);
```

Sets the ::columns property which determines in how many columns the icons are arranged. If `columns` is -1, the number of columns will be chosen automatically to fill the available area.

Parameters

icon_view	a GtkIconView
columns	the number of columns

Since: 2.6

gtk_icon_view_get_columns ()

```
gint
gtk_icon_view_get_columns (GtkIconView *icon_view);
```

Returns the value of the ::columns property.

Parameters

icon_view	a GtkIconView

Returns

the number of columns, or -1

Since: 2.6

---

gtk_icon_view_set_item_width ()

```
void
gtk_icon_view_set_item_width (GtkIconView *icon_view,
                              gint item_width);
```

Sets the ::item-width property which specifies the width to use for each item. If it is set to -1, the icon view will automatically determine a suitable item size.

Parameters

icon_view	a GtkIconView
item_width	the width for each item

Since: 2.6

---

gtk_icon_view_get_item_width ()

```
gint
gtk_icon_view_get_item_width (GtkIconView *icon_view);
```

Returns the value of the ::item-width property.

Parameters

icon_view	a GtkIconView

Returns

the width of a single item, or -1

Since: 2.6

---

gtk_icon_view_set_spacing ()

```
void
gtk_icon_view_set_spacing (GtkIconView *icon_view,
                           gint spacing);
```

Sets the ::spacing property which specifies the space which is inserted between the cells (i.e. the icon and the text) of an item.

Parameters

icon_view	a GtkIconView
spacing	the spacing

Since: 2.6

---

gtk_icon_view_get_spacing ()

```
gint
gtk_icon_view_get_spacing (GtkIconView *icon_view);
```

Returns the value of the ::spacing property.

Parameters

icon_view	a GtkIconView

Returns

the space between cells

Since: 2.6

---

gtk_icon_view_set_row_spacing ()

```
void
gtk_icon_view_set_row_spacing (GtkIconView *icon_view,
                               gint row_spacing);
```

Sets the ::row-spacing property which specifies the space which is inserted between the rows of the icon view.

Parameters

icon_view	a GtkIconView
row_spacing	the row spacing

Since: 2.6

gtk_icon_view_get_row_spacing ()

```
gint
gtk_icon_view_get_row_spacing (GtkIconView *icon_view);
```

Returns the value of the ::row-spacing property.

Parameters

icon_view	a GtkIconView

Returns

the space between rows

Since: 2.6

---

gtk_icon_view_set_column_spacing ()

```
void
gtk_icon_view_set_column_spacing (GtkIconView *icon_view,
                                  gint column_spacing);
```

Sets the ::column-spacing property which specifies the space which is inserted between the columns of the icon view.

Parameters

icon_view	a GtkIconView
column_spacing	the column spacing

Since: 2.6

---

gtk_icon_view_get_column_spacing ()

```
gint
gtk_icon_view_get_column_spacing (GtkIconView *icon_view);
```

Returns the value of the ::column-spacing property.

Parameters

icon_view	a GtkIconView

Returns

the space between columns

Since: 2.6

---

gtk_icon_view_set_margin ()

```
void
gtk_icon_view_set_margin (GtkIconView *icon_view,
                          gint margin);
```

Sets the ::margin property which specifies the space which is inserted at the top, bottom, left and right of the icon view.

Parameters

icon_view	a GtkIconView
margin	the margin

Since: 2.6

---

gtk_icon_view_get_margin ()

```
gint
gtk_icon_view_get_margin (GtkIconView *icon_view);
```

Returns the value of the ::margin property.

Parameters

icon_view	a GtkIconView

Returns

the space at the borders

Since: 2.6

---

gtk_icon_view_set_item_padding ()

```
void
gtk_icon_view_set_item_padding (GtkIconView *icon_view,
                                gint item_padding);
```

Sets the “item-padding” property which specifies the padding around each of the icon view’s items.

Parameters

icon_view	a GtkIconView
item_padding	the item padding

Since: 2.18

gtk_icon_view_get_item_padding ()

```
gint
gtk_icon_view_get_item_padding (GtkIconView *icon_view);
```

Returns the value of the ::item-padding property.

Parameters

icon_view	a GtkIconView

Returns

the padding around items

Since: 2.18

---

gtk_icon_view_set_activate_on_single_click ()

```
void
gtk_icon_view_set_activate_on_single_click
                                (GtkIconView *icon_view,
                                 gboolean single);
```

Causes the “item-activated” signal to be emitted on a single click instead of a double click.

Parameters

icon_view	a GtkIconView
single	TRUE to emit item-activated on a single click

Since: 3.8

---

gtk_icon_view_get_activate_on_single_click ()

```
gboolean
gtk_icon_view_get_activate_on_single_click
                                (GtkIconView *icon_view);
```

Gets the setting set by gtk_icon_view_set_activate_on_single_click().

Parameters

icon_view	a GtkIconView

Returns

TRUE if item-activated will be emitted on a single click

Since: 3.8

---

gtk_icon_view_get_cell_rect ()

```
gboolean
gtk_icon_view_get_cell_rect (GtkIconView *icon_view,
                             GtkTreePath *path,
                             GtkCellRenderer *cell,
                             GdkRectangle *rect);
```

Fills the bounding rectangle in widget coordinates for the cell specified by path and cell . If cell is NULL the main cell area is used.

This function is only valid if icon_view is realized.

Parameters

icon_view	a GtkIconView	
path	a GtkTreePath	
cell	a GtkCellRenderer or NULL.	[allow-none]
rect	rectangle to fill with cell rect.	[out]

Returns

FALSE if there is no such item, TRUE otherwise

Since: 3.6

---

gtk_icon_view_select_path ()

```
void
gtk_icon_view_select_path (GtkIconView *icon_view,
                           GtkTreePath *path);
```

Selects the row at path .

Parameters

icon_view	A GtkIconView.
path	The GtkTreePath to be selected.

Since: 2.6

---

gtk_icon_view_unselect_path ()

```
void
gtk_icon_view_unselect_path (GtkIconView *icon_view,
                             GtkTreePath *path);
```

Unselects the row at `path` .

Parameters

icon_view	A GtkIconView.
path	The GtkTreePath to be unselected.

Since: 2.6

---

gtk_icon_view_path_is_selected ()

```
gboolean
gtk_icon_view_path_is_selected (GtkIconView *icon_view,
                                GtkTreePath *path);
```

Returns `TRUE` if the icon pointed to by `path` is currently selected. If `path` does not point to a valid location, `FALSE` is returned.

Parameters

icon_view	A GtkIconView.
path	A GtkTreePath to check selection on.

Returns

`TRUE` if `path` is selected.

Since: 2.6

---

gtk_icon_view_get_selected_items ()

```
GList *
gtk_icon_view_get_selected_items (GtkIconView *icon_view);
```

Creates a list of paths of all selected items. Additionally, if you are planning on modifying the model after calling this function, you may want to convert the returned list into a list of GtkTreeRowReferences. To do this, you can use `gtk_tree_row_reference_new()`.

To free the return value, use:

```
1
```

```
g_list_free_full (list, (GDestroyNotify)
gtk_tree_path_free);
```

Parameters

icon_view	A GtkIconView.

Returns

A GList containing a GtkTreePath for each selected row.

[element-type GtkTreePath][transfer full]

Since: 2.6

---

gtk_icon_view_select_all ()

```
void
gtk_icon_view_select_all (GtkIconView *icon_view);
```

Selects all the icons. `icon_view` must has its selection mode set to GTK_SELECTION_MULTIPLE.

Parameters

icon_view	A GtkIconView.

Since: 2.6

---

gtk_icon_view_unselect_all ()

```
void
gtk_icon_view_unselect_all (GtkIconView *icon_view);
```

Unselects all the icons.

Parameters

icon_view	A GtkIconView.

Since: 2.6

---

gtk_icon_view_item_activated ()

```
void
gtk_icon_view_item_activated (GtkIconView *icon_view,
                              GtkTreePath *path);
```

Activates the item determined by `path` .

Parameters

icon_view	A GtkIconView
path	The GtkTreePath to be activated

Since: 2.6

---

gtk_icon_view_scroll_to_path ()

```
void
gtk_icon_view_scroll_to_path (GtkIconView *icon_view,
                              GtkTreePath *path,
                              gboolean use_align,
                              gfloat row_align,
                              gfloat col_align);
```

Moves the alignments of `icon_view` to the position specified by `path` . `row_align` determines where the row is placed, and `col_align` determines where `column` is placed. Both are expected to be between 0.0 and 1.0. 0.0 means left/top alignment, 1.0 means right/bottom alignment, 0.5 means center.

If `use_align` is `FALSE`, then the alignment arguments are ignored, and the tree does the minimum amount of work to scroll the item onto the screen. This means that the item will be scrolled to the edge closest to its current position. If the item is currently visible on the screen, nothing is done.

This function only works if the model is set, and `path` is a valid row on the model. If the model changes before the `icon_view` is realized, the centered path will be modified to reflect this change.

Parameters

icon_view	A GtkIconView.
path	The path of the item to move to.
use_align	whether to use alignment arguments, or `FALSE`.
row_align	The vertical alignment of the item specified by `path` .
col_align	The horizontal alignment of the item specified by `path` .

Since: 2.8

---

gtk_icon_view_get_visible_range ()

```
gboolean
gtk_icon_view_get_visible_range (GtkIconView *icon_view,
                                 GtkTreePath **start_path,
                                 GtkTreePath **end_path);
```

Sets `start_path` and `end_path` to be the first and last visible path. Note that there may be invisible paths in between.

Both paths should be freed with `gtk_tree_path_free()` after use.

Parameters

icon_view	A GtkIconView	
start_path	Return location for start of region, or `NULL`.	[out][allow-none]
end_path	Return location for end of region, or `NULL`.	[out][allow-none]

Returns

`TRUE`, if valid paths were placed in `start_path` and `end_path`

Since: 2.8

---

gtk_icon_view_set_tooltip_item ()

```
void
gtk_icon_view_set_tooltip_item (GtkIconView *icon_view,
                                GtkTooltip *tooltip,
                                GtkTreePath *path);
```

Sets the tip area of `tooltip` to be the area covered by the item at `path` . See also `gtk_icon_view_set_tooltip_column()` for a simpler alternative. See also `gtk_tooltip_set_tip_area()`.

Parameters

icon_view	a GtkIconView	
tooltip	a GtkTooltip	
path	a GtkTreePath	

Since: 2.12

---

gtk_icon_view_set_tooltip_cell ()

```
void
gtk_icon_view_set_tooltip_cell (GtkIconView *icon_view,
                                GtkTooltip *tooltip,
                                GtkTreePath *path,
                                GtkCellRenderer *cell);
```

Sets the tip area of `tooltip` to the area which `cell` occupies in the item pointed to by `path` . See also `gtk_tooltip_set_tip_area()`.

See also `gtk_icon_view_set_tooltip_column()` for a simpler alternative.

Parameters

icon_view	a GtkIconView	
tooltip	a GtkTooltip	
path	a GtkTreePath	
cell	a GtkCellRenderer or `NULL`.	[allow-none]

Since: 2.12

---

gtk_icon_view_get_tooltip_context ()

```
gboolean
```

```
gtk_icon_view_get_tooltip_context (GtkIconView *icon_view,
                                   gint *x,
                                   gint *y,
                                   gboolean keyboard_tip,
                                   GtkTreeModel **model,
                                   GtkTreePath **path,
                                   GtkTreeIter *iter);
```

This function is supposed to be used in a "query-tooltip" signal handler for GtkIconView. The `x` , `y` and `keyboard_tip` values which are received in the signal handler, should be passed to this function without modification.

The return value indicates whether there is an icon view item at the given coordinates (`TRUE`) or not (`FALSE`) for mouse tooltips. For keyboard tooltips the item returned will be the cursor item. When `TRUE`, then any of `model` , `path` and `iter` which have been provided will be set to point to that row and the corresponding model. `x` and `y` will always be converted to be relative to `icon_view` 's bin_window if `keyboard_tooltip` is `FALSE`.

Parameters

icon_view	an GtkIconView	
x	the x coordinate (relative to widget coordinates).	[inout]
y	the y coordinate (relative to widget coordinates).	[inout]
keyboard_tip	whether this is a keyboard tooltip or not	
model	a pointer to receive a GtkTreeModel or `NULL`.	[out][allow-none][transfer none]
path	a pointer to receive a GtkTreePath or `NULL`.	[out][allow-none]
iter	a pointer to receive a GtkTreeIter or `NULL`.	[out][allow-none]

Returns

whether or not the given tooltip context points to a item

Since: 2.12

---

gtk_icon_view_set_tooltip_column ()

```
void
gtk_icon_view_set_tooltip_column (GtkIconView *icon_view,
                                  gint column);
```

If you only plan to have simple (text-only) tooltips on full items, you can use this function to have GtkIconView handle these automatically for you. `column` should be set to the column in `icon_view` 's model containing the tooltip texts, or -1 to disable this feature.

When enabled, "has-tooltip" will be set to `TRUE` and `icon_view` will connect a "query-tooltip" signal handler.

Note that the signal handler sets the text with `gtk_tooltip_set_markup()`, so &, <, etc have to be escaped in the text.

Parameters

icon_view	a GtkIconView
column	an integer, which is a valid column number for `icon_view` 's model

Since: 2.12

---

gtk_icon_view_get_tooltip_column ()

```
gint
gtk_icon_view_get_tooltip_column (GtkIconView *icon_view);
```

Returns the column of `icon_view` 's model which is being used for displaying tooltips on `icon_view` 's rows.

Parameters

icon_view	a GtkIconView

Returns

the index of the tooltip column that is currently being used, or -1 if this is disabled.

Since: 2.12

---

gtk_icon_view_get_item_row ()

```
gint
gtk_icon_view_get_item_row (GtkIconView *icon_view,
                            GtkTreePath *path);
```

Gets the row in which the item `path` is currently displayed. Row numbers start at 0.

Parameters

icon_view	a GtkIconView
path	the GtkTreePath of the item

Returns

The row in which the item is displayed

Since: 2.22

---

gtk_icon_view_get_item_column ()

```
gint
gtk_icon_view_get_item_column (GtkIconView *icon_view,
                               GtkTreePath *path);
```

Gets the column in which the item `path` is currently displayed. Column numbers start at 0.

Parameters

icon_view	a GtkIconView
path	the GtkTreePath of the item

Returns

The column in which the item is displayed

Since: 2.22

---

gtk_icon_view_enable_model_drag_source ()

```
void
gtk_icon_view_enable_model_drag_source
                               (GtkIconView *icon_view,
                                GdkModifierType start_button_mask,
                                const GtkTargetEntry *targets,
                                gint n_targets,
                                GdkDragAction actions);
```

Turns `icon_view` into a drag source for automatic DND. Calling this method sets "reorderable" to `FALSE`.

Parameters

icon_view	a GtkIconView	
start_button_mask	Mask of allowed buttons to start drag	
targets	the table of targets that the drag will support.	[array length=n_targets]
n_targets	the number of items in `targets`	
actions	the bitmask of possible actions for a drag from this widget	

Since: 2.8

---

gtk_icon_view_enable_model_drag_dest ()

```
void
gtk_icon_view_enable_model_drag_dest (GtkIconView *icon_view,
                                      const GtkTargetEntry *targets,
                                      gint n_targets,
                                      GdkDragAction actions);
```

Turns `icon_view` into a drop destination for automatic DND. Calling this method sets "reorderable" to `FALSE`.

Parameters

icon_view	a GtkIconView
targets	the table of targets that the drag will support. [array length=n_targets]
n_targets	the number of items in `targets`
actions	the bitmask of possible actions for a drag to this widget

Since: 2.8

---

gtk_icon_view_unset_model_drag_source ()

```
void
gtk_icon_view_unset_model_drag_source (GtkIconView *icon_view);
```

Undoes the effect of `gtk_icon_view_enable_model_drag_source()`. Calling this method sets "reorderable" to `FALSE`.

Parameters

icon_view	a GtkIconView

Since: 2.8

---

gtk_icon_view_unset_model_drag_dest ()

```
void
gtk_icon_view_unset_model_drag_dest (GtkIconView *icon_view);
```

Undoes the effect of `gtk_icon_view_enable_model_drag_dest()`. Calling this method sets "reorderable" to `FALSE`.

Parameters

icon_view	a GtkIconView

Since: 2.8

---

gtk_icon_view_set_reorderable ()

```
void
gtk_icon_view_set_reorderable (GtkIconView *icon_view,
                               gboolean reorderable);
```

This function is a convenience function to allow you to reorder models that support the GtkTreeDragSourceIface and the GtkTreeDragDestIface. Both GtkTreeStore and GtkListStore support these. If `reorderable` is `TRUE`, then the user can reorder the model by dragging and dropping rows. The developer can listen to these changes by connecting to the model's row_inserted and row_deleted signals. The reordering is

implemented by setting up the icon view as a drag source and destination. Therefore, drag and drop can not be used in a reorderable view for any other purpose.

This function does not give you any degree of control over the order -- any reordering is allowed. If more control is needed, you should probably handle drag and drop manually.

Parameters

icon_view	A GtkIconView.
reorderable	TRUE, if the list of items can be reordered.

Since: 2.8

---

gtk_icon_view_get_reorderable ()

```
gboolean
gtk_icon_view_get_reorderable (GtkIconView *icon_view);
```

Retrieves whether the user can reorder the list via drag-and-drop. See gtk_icon_view_set_reorderable().

Parameters

icon_view	a GtkIconView

Returns

TRUE if the list can be reordered.

Since: 2.8

---

gtk_icon_view_set_drag_dest_item ()

```
void
gtk_icon_view_set_drag_dest_item (GtkIconView *icon_view,
                                  GtkTreePath *path,
                                  GtkIconViewDropPosition pos);
```

Sets the item that is highlighted for feedback.

Parameters

icon_view	a GtkIconView	
path	The path of the item to highlight, or NULL.	[allow-none]
pos	Specifies where to drop, relative to the item	

Since: 2.8

---

gtk_icon_view_get_drag_dest_item ()

```
void
gtk_icon_view_get_drag_dest_item (GtkIconView *icon_view,
                                  GtkTreePath **path,
                                  GtkIconViewDropPosition *pos);
```

Gets information about the item that is highlighted for feedback.

Parameters

icon_view	a GtkIconView	
path	Return location for the path of the highlighted item, or `NULL`.	[out][allow-none]
pos	Return location for the drop position, or `NULL`.	[out][allow-none]

Since: 2.8

---

gtk_icon_view_get_dest_item_at_pos ()

```
gboolean
gtk_icon_view_get_dest_item_at_pos (GtkIconView *icon_view,
                                    gint drag_x,
                                    gint drag_y,
                                    GtkTreePath **path,
                                    GtkIconViewDropPosition *pos);
```

Determines the destination item for a given position.

Parameters

icon_view	a GtkIconView	
drag_x	the position to determine the destination item for	
drag_y	the position to determine the destination item for	
path	Return location for the path of the item, or `NULL`.	[out][allow-none]
pos	Return location for the drop position, or `NULL`.	[out][allow-none]

Returns

whether there is an item at the given position.

Since: 2.8

---

gtk_icon_view_create_drag_icon ()

```
cairo_surface_t *
gtk_icon_view_create_drag_icon (GtkIconView *icon_view,
                                GtkTreePath *path);
```

Creates a cairo_surface_t representation of the item at `path` . This image is used for a drag icon.

Parameters

icon_view	a GtkIconView
path	a GtkTreePath in `icon_view`

Returns

a newly-allocated surface of the drag icon.

[transfer full]

Since: 2.8

Types and Values

struct GtkIconView

```
struct GtkIconView;
```

---

enum GtkIconViewDropPosition

An enum for determining where a dropped item goes.

Members

GTK_ICON_VIEW_NO_DROP	no drop possible
GTK_ICON_VIEW_DROP_INTO	dropped item replaces the item
GTK_ICON_VIEW_DROP_LEFT	droppped item is inserted to the left
GTK_ICON_VIEW_DROP_RIGHT	dropped item is inserted to the right
GTK_ICON_VIEW_DROP_ABOVE	dropped item is inserted above
GTK_ICON_VIEW_DROP_BELOW	dropped item is inserted below

Property Details

The “activate-on-single-click” property

```
“activate-on-single-click” gboolean
```

The activate-on-single-click property specifies whether the "item-activated" signal will be emitted after a single click.

Flags: Read / Write

Default value: FALSE

Since: 3.8

The “cell-area” property

```
“cell-area”                GtkCellArea *
```

The GtkCellArea used to layout cell renderers for this view.

If no area is specified when creating the icon view with `gtk_icon_view_new_with_area()` a GtkCellAreaBox will be used.

Flags: Read / Write / Construct Only

Since: 3.0

The “column-spacing” property

```
“column-spacing”           gint
```

The column-spacing property specifies the space which is inserted between the columns of the icon view.

Flags: Read / Write

Allowed values: >= 0

Default value: 6

Since: 2.6

The “columns” property

```
“columns”                  gint
```

The columns property contains the number of the columns in which the items should be displayed. If it is -1, the number of columns will be chosen automatically to fill the available area.

Flags: Read / Write

Allowed values: >= -1

Default value: -1

Since: 2.6

The “item-orientation” property

“item-orientation” GtkOrientation

The item-orientation property specifies how the cells (i.e. the icon and the text) of the item are positioned relative to each other.

Flags: Read / Write

Default value: GTK_ORIENTATION_VERTICAL

Since: 2.6

The “item-padding” property

“item-padding” gint

The item-padding property specifies the padding around each of the icon view's item.

Flags: Read / Write

Allowed values: >= 0

Default value: 6

Since: 2.18

The “item-width” property

“item-width” gint

The item-width property specifies the width to use for each item. If it is set to -1, the icon view will automatically determine a suitable item size.

Flags: Read / Write

Allowed values: >= -1

Default value: -1

Since: 2.6

The “margin” property

“margin” gint

The margin property specifies the space which is inserted at the edges of the icon view.

Flags: Read / Write

Allowed values: >= 0

Default value: 6

Since: 2.6

The “markup-column” property

“markup-column” gint

The ::markup-column property contains the number of the model column containing markup information to be displayed. The markup column must be of type G_TYPE_STRING. If this property and the :text-column property are both set to column numbers, it overrides the text column. If both are set to -1, no texts are displayed.

Flags: Read / Write

Allowed values: >= -1

Default value: -1

Since: 2.6

The “model” property

“model” GtkTreeModel *

The model for the icon view.

Flags: Read / Write

The “pixbuf-column” property

“pixbuf-column” gint

The ::pixbuf-column property contains the number of the model column containing the pixbufs which are displayed. The pixbuf column must be of type GDK_TYPE_PIXBUF. Setting this property to -1 turns off the display of pixbufs.

Flags: Read / Write

Allowed values: >= -1

Default value: -1

Since: 2.6

The “reorderable” property

“reorderable” gboolean

The reorderable property specifies if the items can be reordered by DND.

Flags: Read / Write

Default value: FALSE

Since: 2.8

The “row-spacing” property

“row-spacing” gint

The row-spacing property specifies the space which is inserted between the rows of the icon view.

Flags: Read / Write

Allowed values: >= 0

Default value: 6

Since: 2.6

The “selection-mode” property

“selection-mode” GtkSelectionMode

The ::selection-mode property specifies the selection mode of icon view. If the mode is GTK_SELECTION_MULTIPLE, rubberband selection is enabled, for the other modes, only keyboard selection is possible.

Flags: Read / Write

Default value: GTK_SELECTION_SINGLE

Since: 2.6

The “spacing” property

“spacing” gint

The spacing property specifies the space which is inserted between the cells (i.e. the icon and the text) of an item.

Flags: Read / Write

Allowed values: >= 0

Default value: 0

Since: 2.6

The “text-column” property

“text-column” gint

The ::text-column property contains the number of the model column containing the texts which are displayed. The text column must be of type G_TYPE_STRING. If this property and the :markup-column property are both set to -1, no texts are displayed.

Flags: Read / Write

Allowed values: >= -1

Default value: -1

Since: 2.6

The “tooltip-column” property

“tooltip-column” gint

The column in the model containing the tooltip texts for the items.

Flags: Read / Write

Allowed values: >= -1

Default value: -1

Style Property Details

The “selection-box-alpha” style property

“selection-box-alpha” guchar

The opacity of the selection box.

GtkIconView:selection-box-alpha has been deprecated since version 3.20 and should not be used in newly-written code.

The opacity of the selection box is determined by CSS; the value of this style property is ignored.

Flags: Read

Default value: 64

The “selection-box-color” style property

“selection-box-color” GdkColor *

The color of the selection box.

GtkIconView:selection-box-color has been deprecated since version 3.20 and should not be used in newly-written code.

The color of the selection box is determined by CSS; the value of this style property is ignored.

Flags: Read

Signal Details

The “activate-cursor-item” signal

```
gboolean
user_function (GtkIconView *iconview,
               gpointer     user_data)
```

A keybinding signal which gets emitted when the user activates the currently focused item.

Applications should not connect to it, but may emit it with `g_signal_emit_by_name()` if they need to control activation programmatically.

The default bindings for this signal are Space, Return and Enter.

Parameters

iconview	the object on which the signal is emitted
user_data	user data set when the signal handler was connected.

Flags: Action

The “item-activated” signal

```
void
user_function (GtkIconView *iconview,
               GtkTreePath *path,
               gpointer     user_data)
```

The ::item-activated signal is emitted when the method `gtk_icon_view_item_activated()` is called, when the user double clicks an item with the "activate-on-single-click" property set to `FALSE`, or when the user single clicks an item when the "activate-on-single-click" property set to `TRUE`. It is also emitted when a non-editable item is selected and one of the keys: Space, Return or Enter is pressed.

Parameters

iconview	the object on which the signal is emitted
path	the GtkTreePath for the activated item
user_data	user data set when the signal handler was connected.

Flags: Run Last

The “move-cursor” signal

```
gboolean
user_function (GtkIconView    *iconview,
               GtkMovementStep step,
               gint            count,
               gpointer        user_data)
```

The ::move-cursor signal is a keybinding signal which gets emitted when the user initiates a cursor movement.

Applications should not connect to it, but may emit it with `g_signal_emit_by_name()` if they need to control the cursor programmatically.

The default bindings for this signal include

- Arrow keys which move by individual steps
- Home/End keys which move to the first/last item
- PageUp/PageDown which move by "pages" All of these will extend the selection when combined with the Shift modifier.

Parameters

iconview	the object which received the signal
step	the granularity of the move, as a GtkMovementStep
count	the number of `step` units to move
user_data	user data set when the signal handler was connected.

Flags: Action

The `"select-all"` signal

```
void
user_function (GtkIconView *iconview,
               gpointer     user_data)
```

A keybinding signal which gets emitted when the user selects all items.

Applications should not connect to it, but may emit it with `g_signal_emit_by_name()` if they need to control selection programmatically.

The default binding for this signal is Ctrl-a.

Parameters

iconview	the object on which the signal is emitted
user_data	user data set when the signal handler was connected.

Flags: Action

The `"select-cursor-item"` signal

```
void
user_function (GtkIconView *iconview,
               gpointer     user_data)
```

A keybinding signal which gets emitted when the user selects the item that is currently focused.

Applications should not connect to it, but may emit it with `g_signal_emit_by_name()` if they need to control

selection programmatically.

There is no default binding for this signal.

Parameters

iconview	the object on which the signal is emitted
user_data	user data set when the signal handler was connected.

Flags: Action

The `"selection-changed"` signal

```
void
user_function (GtkIconView *iconview,
               gpointer     user_data)
```

The ::selection-changed signal is emitted when the selection (i.e. the set of selected items) changes.

Parameters

iconview	the object on which the signal is emitted
user_data	user data set when the signal handler was connected.

Flags: Run First

The `"toggle-cursor-item"` signal

```
void
user_function (GtkIconView *iconview,
               gpointer     user_data)
```

A keybinding signal which gets emitted when the user toggles whether the currently focused item is selected or not. The exact effect of this depend on the selection mode.

Applications should not connect to it, but may emit it with `g_signal_emit_by_name()` if they need to control selection programmatically.

There is no default binding for this signal is Ctrl-Space.

Parameters

iconview	the object on which the signal is emitted
user_data	user data set when the signal handler was connected.

Flags: Action

The "unselect-all" signal

```
void
user_function (GtkIconView *iconview,
               gpointer     user_data)
```

A keybinding signal which gets emitted when the user unselects all items.

Applications should not connect to it, but may emit it with `g_signal_emit_by_name()` if they need to control selection programmatically.

The default binding for this signal is Ctrl-Shift-a.

Parameters

iconview	the object on which the signal is emitted
user_data	user data set when the signal handler was connected.

Flags: Action

GtkTreeSortable

GtkTreeSortable — The interface for sortable models used by GtkTreeView

Functions

gint	(*GtkTreeIterCompareFunc) ()
void	gtk_tree_sortable_sort_column_changed ()
gboolean	gtk_tree_sortable_get_sort_column_id ()
void	gtk_tree_sortable_set_sort_column_id ()
void	gtk_tree_sortable_set_sort_func ()
void	gtk_tree_sortable_set_default_sort_func ()
gboolean	gtk_tree_sortable_has_default_sort_func ()

Signals

void	sort-column-changed	Run Last

Types and Values

	GtkTreeSortable
struct	GtkTreeSortableIface
#define	GTK_TREE_SORTABLE_DEFAULT_SORT_COLUMN_ID

#define	GTK_TREE_SORTABLE_UNSORTED_SORT_COLUMN_ID

Object Hierarchy

```
GInterface
└── GtkTreeSortable
```

Prerequisites

GtkTreeSortable requires GtkTreeModel and GObject.

Known Implementations

GtkTreeSortable is implemented by GtkListStore, GtkTreeModelSort and GtkTreeStore.

Includes

```
#include <gtk/gtk.h>
```

Description

GtkTreeSortable is an interface to be implemented by tree models which support sorting. The GtkTreeView uses the methods provided by this interface to sort the model.

Functions

GtkTreeIterCompareFunc ()

```
gint
(*GtkTreeIterCompareFunc) (GtkTreeModel *model,
                           GtkTreeIter *a,
                           GtkTreeIter *b,
                           gpointer user_data);
```

A GtkTreeIterCompareFunc should return a negative integer, zero, or a positive integer if `a` sorts before `b` , `a` sorts with `b` , or `a` sorts after `b` respectively. If two iters compare as equal, their order in the sorted model is undefined. In order to ensure that the GtkTreeSortable behaves as expected, the GtkTreeIterCompareFunc must define a partial order on the model, i.e. it must be reflexive, antisymmetric and transitive.

For example, if `model` is a product catalogue, then a compare function for the “price” column could be one which returns `price_of(@a) - price_of(@b)`.

Parameters

model	The GtkTreeModel the comparison is within

a	A GtkTreeIter in `model`
b	Another GtkTreeIter in `model`
user_data	Data passed when the compare func is assigned e.g. by `gtk_tree_sortable_set_sort_func()`

Returns

a negative integer, zero or a positive integer depending on whether `a` sorts before, with or after `b`

---

gtk_tree_sortable_sort_column_changed ()

```
void
gtk_tree_sortable_sort_column_changed (GtkTreeSortable *sortable);
```

Emits a "sort-column-changed" signal on `sortable`.

Parameters

sortable	A GtkTreeSortable

---

gtk_tree_sortable_get_sort_column_id ()

```
gboolean
gtk_tree_sortable_get_sort_column_id (GtkTreeSortable *sortable,
                                      gint *sort_column_id,
                                      GtkSortType *order);
```

Fills in `sort_column_id` and `order` with the current sort column and the order. It returns `TRUE` unless the `sort_column_id` is `GTK_TREE_SORTABLE_DEFAULT_SORT_COLUMN_ID` or `GTK_TREE_SORTABLE_UNSORTED_SORT_COLUMN_ID`.

Parameters

sortable	A GtkTreeSortable	
sort_column_id	The sort column id to be filled in.	[out]
order	The GtkSortType to be filled in.	[out]

Returns

`TRUE` if the sort column is not one of the special sort column ids.

---

gtk_tree_sortable_set_sort_column_id ()

```
void
gtk_tree_sortable_set_sort_column_id (GtkTreeSortable *sortable,
```

```
                                gint sort_column_id,
                                GtkSortType order);
```

Sets the current sort column to be `sort_column_id` . The `sortable` will resort itself to reflect this change, after emitting a "sort-column-changed" signal. `sort_column_id` may either be a regular column id, or one of the following special values:

- `GTK_TREE_SORTABLE_DEFAULT_SORT_COLUMN_ID`: the default sort function will be used, if it is set
- `GTK_TREE_SORTABLE_UNSORTED_SORT_COLUMN_ID`: no sorting will occur

Parameters

sortable	A GtkTreeSortable	
sort_column_id	the sort column id to set	
order	The sort order of the column	

---

gtk_tree_sortable_set_sort_func ()

```
void
gtk_tree_sortable_set_sort_func (GtkTreeSortable *sortable,
                                 gint sort_column_id,
                                 GtkTreeIterCompareFunc sort_func,
                                 gpointer user_data,
                                 GDestroyNotify destroy);
```

Sets the comparison function used when sorting to be `sort_func` . If the current sort column id of `sortable` is the same as `sort_column_id` , then the model will sort using this function.

Parameters

sortable	A GtkTreeSortable	
sort_column_id	the sort column id to set the function for	
sort_func	The comparison function	
user_data	User data to pass to `sort_func` , or `NULL`.	[allow-none]
destroy	Destroy notifier of `user_data` , or `NULL`.	[allow-none]

---

gtk_tree_sortable_set_default_sort_func ()

```
void
gtk_tree_sortable_set_default_sort_func
                                (GtkTreeSortable *sortable,
                                 GtkTreeIterCompareFunc sort_func,
                                 gpointer user_data,
                                 GDestroyNotify destroy);
```

Sets the default comparison function used when sorting to be `sort_func` . If the current sort column id of `sortable` is `GTK_TREE_SORTABLE_DEFAULT_SORT_COLUMN_ID`, then the model will sort using this function.

If `sort_func` is `NULL`, then there will be no default comparison function. This means that once the model has

been sorted, it can't go back to the default state. In this case, when the current sort column id of `sortable` is `GTK_TREE_SORTABLE_DEFAULT_SORT_COLUMN_ID`, the model will be unsorted.

Parameters

sortable	A GtkTreeSortable	
sort_func	The comparison function	
user_data	User data to pass to `sort_func` , or `NULL`.	[allow-none]
destroy	Destroy notifier of `user_data` , or `NULL`.	[allow-none]

---

gtk_tree_sortable_has_default_sort_func ()

```
gboolean
gtk_tree_sortable_has_default_sort_func
                                (GtkTreeSortable *sortable);
```

Returns `TRUE` if the model has a default sort function. This is used primarily by GtkTreeViewColumns in order to determine if a model can go back to the default state, or not.

Parameters

sortable	A GtkTreeSortable

Returns

`TRUE`, if the model has a default sort function

Types and Values

GtkTreeSortable

```
typedef struct _GtkTreeSortable GtkTreeSortable;
```

---

struct GtkTreeSortableIface

```
struct GtkTreeSortableIface {
  /* signals */
  void     (* sort_column_changed)   (GtkTreeSortable        *sortable);

  /* virtual table */
  gboolean (* get_sort_column_id)    (GtkTreeSortable        *sortable,
                                      gint                   *sort_column_id,
                                      GtkSortType            *order);
  void     (* set_sort_column_id)    (GtkTreeSortable        *sortable,
```

```
                                gint                    sort_column_id,
                                GtkSortType             order);
  void     (* set_sort_func)         (GtkTreeSortable       *sortable,
                                gint                    sort_column_id,
                                GtkTreeIterCompareFunc  sort_func,
                                gpointer                user_data,
                                GDestroyNotify          destroy);
  void     (* set_default_sort_func) (GtkTreeSortable       *sortable,
                                GtkTreeIterCompareFunc  sort_func,
                                gpointer                user_data,
                                GDestroyNotify          destroy);
  gboolean (* has_default_sort_func) (GtkTreeSortable       *sortable);
};
```

Members

`sort_column_changed` ()	Signal emitted when the sort column or sort order of sortable is changed.
`get_sort_column_id` ()	Fills in sort_column_id and order with the current sort column and the order.
`set_sort_column_id` ()	Sets the current sort column to be sort_column_id.
`set_sort_func` ()	Sets the comparison function used when sorting to be sort_func.
`set_default_sort_func` ()	Sets the default comparison function used when sorting to be sort_func.
`has_default_sort_func` ()	`TRUE` if the model has a default sort function.

GTK_TREE_SORTABLE_DEFAULT_SORT_COLUMN_ID

```
#define GTK_TREE_SORTABLE_DEFAULT_SORT_COLUMN_ID (-1)
```

The GTK_TREE_SORTABLE_DEFAULT_SORT_COLUMN_ID can be used to make a GtkTreeSortable use the default sort function.

See also `gtk_tree_sortable_set_sort_column_id()`

GTK_TREE_SORTABLE_UNSORTED_SORT_COLUMN_ID

```
#define GTK_TREE_SORTABLE_UNSORTED_SORT_COLUMN_ID (-2)
```

The GTK_TREE_SORTABLE_DEFAULT_SORT_COLUMN_ID can be used to make a GtkTreeSortable use no sorting.

See also `gtk_tree_sortable_set_sort_column_id()`

Signal Details

The "sort-column-changed" signal

```
void
user_function (GtkTreeSortable *sortable,
               gpointer         user_data)
```

The ::sort-column-changed signal is emitted when the sort column or sort order of `sortable` is changed. The signal is emitted before the contents of `sortable` are resorted.

Parameters

sortable	the object on which the signal is emitted
user_data	user data set when the signal handler was connected.

Flags: Run Last

See Also

GtkTreeModel, GtkTreeView

GtkTreeModelSort

GtkTreeModelSort — A GtkTreeModel which makes an underlying tree model sortable

Functions

GtkTreeModel *	gtk_tree_model_sort_new_with_model ()
GtkTreeModel *	gtk_tree_model_sort_get_model ()
GtkTreePath *	gtk_tree_model_sort_convert_child_path_to_path ()
gboolean	gtk_tree_model_sort_convert_child_iter_to_iter ()
GtkTreePath *	gtk_tree_model_sort_convert_path_to_child_path ()
void	gtk_tree_model_sort_convert_iter_to_child_iter ()
void	gtk_tree_model_sort_reset_default_sort_func ()
void	gtk_tree_model_sort_clear_cache ()
gboolean	gtk_tree_model_sort_iter_is_valid ()

Properties

GtkTreeModel *	model	Read / Write / Construct Only

Types and Values

struct	GtkTreeModelSort

Object Hierarchy

```
GObject
└── GtkTreeModelSort
```

Implemented Interfaces

GtkTreeModelSort implements GtkTreeModel, GtkTreeSortable and GtkTreeDragSource.

Includes

```
#include <gtk/gtk.h>
```

Description

The GtkTreeModelSort is a model which implements the GtkTreeSortable interface. It does not hold any data itself, but rather is created with a child model and proxies its data. It has identical column types to this child model, and the changes in the child are propagated. The primary purpose of this model is to provide a way to sort a different model without modifying it. Note that the sort function used by GtkTreeModelSort is not guaranteed to be stable.

The use of this is best demonstrated through an example. In the following sample code we create two GtkTreeView widgets each with a view of the same data. As the model is wrapped here by a GtkTreeModelSort, the two GtkTreeViews can each sort their view of the data without affecting the other. By contrast, if we simply put the same model in each widget, then sorting the first would sort the second.

Using a GtkTreeModelSort

```
{
  GtkTreeView *tree_view1;
  GtkTreeView *tree_view2;
  GtkTreeModel *sort_model1;
  GtkTreeModel *sort_model2;
  GtkTreeModel *child_model;

  // get the child model
  child_model = get_my_model ();

  // Create the first tree
  sort_model1 =
gtk_tree_model_sort_new_with_model
(child_model);
  tree_view1 = gtk_tree_view_new_with_model
(sort_model1);

  // Create the second tree
  sort_model2 =
gtk_tree_model_sort_new_with_model
(child_model);
  tree_view2 = gtk_tree_view_new_with_model
(sort_model2);

  // Now we can sort the two models
```

```
independently
  gtk_tree_sortable_set_sort_column_id
(GTK_TREE_SORTABLE (sort_model1),

COLUMN_1, GTK_SORT_ASCENDING);
  gtk_tree_sortable_set_sort_column_id
(GTK_TREE_SORTABLE (sort_model2),

COLUMN_1, GTK_SORT_DESCENDING);
}
```

To demonstrate how to access the underlying child model from the sort model, the next example will be a callback for the GtkTreeSelection "changed" signal. In this callback, we get a string from COLUMN_1 of the model. We then modify the string, find the same selected row on the child model, and change the row there.

Accessing the child model of in a selection changed callback

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
```

```
void
selection_changed (GtkTreeSelection
*selection, gpointer data)
{
  GtkTreeModel *sort_model = NULL;
  GtkTreeModel *child_model;
  GtkTreeIter sort_iter;
  GtkTreeIter child_iter;
  char *some_data = NULL;
  char *modified_data;

  // Get the current selected row and the
model.
  if (! gtk_tree_selection_get_selected
(selection,

&sort_model,

&sort_iter))
    return;

  // Look up the current value on the
selected row and get
  // a new value to change it to.
  gtk_tree_model_get (GTK_TREE_MODEL
(sort_model), &sort_iter,
                      COLUMN_1, &some_data,
                      -1);

  modified_data = change_the_data
(some_data);
  g_free (some_data);

  // Get an iterator on the child model,
instead of the sort model.

gtk_tree_model_sort_convert_iter_to_child_ite
r (GTK_TREE_MODEL_SORT (sort_model),

&child_iter,

&sort_iter);

  // Get the child model and change the value
of the row. In this
```

```
  // example, the child model is a
GtkListStore. It could be any other
  // type of model, though.
  child_model = gtk_tree_model_sort_get_model
(GTK_TREE_MODEL_SORT (sort_model));
  gtk_list_store_set (GTK_LIST_STORE
(child_model), &child_iter,
                      COLUMN_1,
&modified_data,
                      -1);
  g_free (modified_data);
}
```

Functions

gtk_tree_model_sort_new_with_model ()

```
GtkTreeModel *
gtk_tree_model_sort_new_with_model (GtkTreeModel *child_model);
```

Creates a new GtkTreeModel, with `child_model` as the child model.

Parameters

child_model	A GtkTreeModel

Returns

A new GtkTreeModel.

[transfer full]

---

gtk_tree_model_sort_get_model ()

```
GtkTreeModel *
gtk_tree_model_sort_get_model (GtkTreeModelSort *tree_model);
```

Returns the model the GtkTreeModelSort is sorting.

Parameters

tree_model	a GtkTreeModelSort

Returns

the "child model" being sorted.

[transfer none]

---

gtk_tree_model_sort_convert_child_path_to_path ()

```
GtkTreePath *
gtk_tree_model_sort_convert_child_path_to_path
                                (GtkTreeModelSort *tree_model_sort,
                                 GtkTreePath *child_path);
```

Converts `child_path` to a path relative to `tree_model_sort` . That is, `child_path` points to a path in the child model. The returned path will point to the same row in the sorted model. If `child_path` isn't a valid path on the child model, then `NULL` is returned.

Parameters

tree_model_sort	A GtkTreeModelSort	
child_path	A GtkTreePath to convert	

Returns

A newly allocated GtkTreePath, or `NULL`.

[nullable][transfer full]

---

gtk_tree_model_sort_convert_child_iter_to_iter ()

```
gboolean
gtk_tree_model_sort_convert_child_iter_to_iter
                                (GtkTreeModelSort *tree_model_sort,
                                 GtkTreeIter *sort_iter,
                                 GtkTreeIter *child_iter);
```

Sets `sort_iter` to point to the row in `tree_model_sort` that corresponds to the row pointed at by `child_iter` . If `sort_iter` was not set, `FALSE` is returned. Note: a boolean is only returned since 2.14.

Parameters

tree_model_sort	A GtkTreeModelSort	
sort_iter	An uninitialized GtkTreeIter.	[out]
child_iter	A valid GtkTreeIter pointing to a row on the child model	

Returns

`TRUE`, if `sort_iter` was set, i.e. if `sort_iter` is a valid iterator pointer to a visible row in the child model.

---

gtk_tree_model_sort_convert_path_to_child_path ()

```
GtkTreePath *
gtk_tree_model_sort_convert_path_to_child_path
                                (GtkTreeModelSort *tree_model_sort,
                                 GtkTreePath *sorted_path);
```

Converts `sorted_path` to a path on the child model of `tree_model_sort` . That is, `sorted_path` points to a location in `tree_model_sort` . The returned path will point to the same location in the model not being sorted. If `sorted_path` does not point to a location in the child model, `NULL` is returned.

Parameters

tree_model_sort	A GtkTreeModelSort
sorted_path	A GtkTreePath to convert

Returns

A newly allocated GtkTreePath, or `NULL`.

[nullable][transfer full]

---

gtk_tree_model_sort_convert_iter_to_child_iter ()

```
void
gtk_tree_model_sort_convert_iter_to_child_iter
                                (GtkTreeModelSort *tree_model_sort,
                                 GtkTreeIter *child_iter,
                                 GtkTreeIter *sorted_iter);
```

Sets `child_iter` to point to the row pointed to by `sorted_iter` .

Parameters

tree_model_sort	A GtkTreeModelSort	
child_iter	An uninitialized GtkTreeIter.	[out]
sorted_iter	A valid GtkTreeIter pointing to a row on `tree_model_sort` .	

---

gtk_tree_model_sort_reset_default_sort_func ()

```
void
gtk_tree_model_sort_reset_default_sort_func
                                (GtkTreeModelSort *tree_model_sort);
```

This resets the default sort function to be in the “unsorted” state. That is, it is in the same order as the child model. It will re-sort the model to be in the same order as the child model only if the GtkTreeModelSort is in “unsorted” state.

Parameters

tree_model_sort	A GtkTreeModelSort

---

gtk_tree_model_sort_clear_cache ()

```
void
gtk_tree_model_sort_clear_cache (GtkTreeModelSort *tree_model_sort);
```

This function should almost never be called. It clears the `tree_model_sort` of any cached iterators that haven't been reffed with `gtk_tree_model_ref_node()`. This might be useful if the child model being sorted is static (and doesn't change often) and there has been a lot of unreffed access to nodes. As a side effect of this function, all unreffed iters will be invalid.

Parameters

tree_model_sort	A GtkTreeModelSort

---

gtk_tree_model_sort_iter_is_valid ()

```
gboolean
gtk_tree_model_sort_iter_is_valid (GtkTreeModelSort *tree_model_sort,
                                   GtkTreeIter *iter);
```

This function is slow. Only use it for debugging and/or testing purposes.

Checks if the given iter is a valid iter for this GtkTreeModelSort.

Parameters

tree_model_sort	A GtkTreeModelSort.
iter	A GtkTreeIter.

Returns

`TRUE` if the iter is valid, `FALSE` if the iter is invalid.

Since: 2.2

Types and Values

struct GtkTreeModelSort

```
struct GtkTreeModelSort;
```

Property Details

The "model" property

```
  "model"                    GtkTreeModel *
```

The model for the TreeModelSort to sort.

Flags: Read / Write / Construct Only

See Also

GtkTreeModel, GtkListStore, GtkTreeStore, GtkTreeSortable, GtkTreeModelFilter

---

GtkTreeModelFilter

GtkTreeModelFilter — A GtkTreeModel which hides parts of an underlying tree model

Functions

gboolean	(*GtkTreeModelFilterVisibleFunc) ()
void	(*GtkTreeModelFilterModifyFunc) ()
GtkTreeModel *	gtk_tree_model_filter_new ()
void	gtk_tree_model_filter_set_visible_func ()
void	gtk_tree_model_filter_set_modify_func ()
void	gtk_tree_model_filter_set_visible_column ()
GtkTreeModel *	gtk_tree_model_filter_get_model ()
gboolean	gtk_tree_model_filter_convert_child_iter_to_iter ()
void	gtk_tree_model_filter_convert_iter_to_child_iter ()
GtkTreePath *	gtk_tree_model_filter_convert_child_path_to_path ()
GtkTreePath *	gtk_tree_model_filter_convert_path_to_child_path ()
void	gtk_tree_model_filter_refilter ()
void	gtk_tree_model_filter_clear_cache ()

Properties

GtkTreeModel *	child-model	Read / Write / Construct Only
GtkTreePath *	virtual-root	Read / Write / Construct Only

Types and Values

struct	GtkTreeModelFilter

Object Hierarchy

```
GObject
└── GtkTreeModelFilter
```

Implemented Interfaces

GtkTreeModelFilter implements GtkTreeModel and GtkTreeDragSource.

Includes

```
#include <gtk/gtk.h>
```

Description

A GtkTreeModelFilter is a tree model which wraps another tree model, and can do the following things:

- Filter specific rows, based on data from a “visible column”, a column storing booleans indicating whether the row should be filtered or not, or based on the return value of a “visible function”, which gets a model, iter and user_data and returns a boolean indicating whether the row should be filtered or not.
- Modify the “appearance” of the model, using a modify function. This is extremely powerful and allows for just changing some values and also for creating a completely different model based on the given child model.
- Set a different root node, also known as a “virtual root”. You can pass in a GtkTreePath indicating the root node for the filter at construction time.

The basic API is similar to GtkTreeModelSort. For an example on its usage, see the section on GtkTreeModelSort.

When using GtkTreeModelFilter, it is important to realize that GtkTreeModelFilter maintains an internal cache of all nodes which are visible in its clients. The cache is likely to be a subtree of the tree exposed by the child model. GtkTreeModelFilter will not cache the entire child model when unnecessary to not compromise the caching mechanism that is exposed by the reference counting scheme. If the child model implements reference counting, unnecessary signals may not be emitted because of reference counting rule 3, see the GtkTreeModel documentation. (Note that e.g. GtkTreeStore does not implement reference counting and will always emit all signals, even when the receiving node is not visible).

Because of this, limitations for possible visible functions do apply. In general, visible functions should only use data or properties from the node for which the visibility state must be determined, its siblings or its parents. Usually, having a dependency on the state of any child node is not possible, unless references are taken on these explicitly. When no such reference exists, no signals may be received for these child nodes (see reference couting rule number 3 in the GtkTreeModel section).

Determining the visibility state of a given node based on the state of its child nodes is a frequently occurring use case. Therefore, GtkTreeModelFilter explicitly supports this. For example, when a node does not have any children, you might not want the node to be visible. As soon as the first row is added to the node’s child level (or the last row removed), the node’s visibility should be updated.

This introduces a dependency from the node on its child nodes. In order to accommodate this, GtkTreeModelFilter must make sure the necessary signals are received from the child model. This is achieved by building, for all nodes which are exposed as visible nodes to GtkTreeModelFilter's clients, the child level (if any) and take a reference on the first node in this level. Furthermore, for every row-inserted, row-changed or row-deleted signal (also these which were not handled because the node was not cached), GtkTreeModelFilter will check if the visibility state of any parent node has changed.

Beware, however, that this explicit support is limited to these two cases. For example, if you want a node to be visible only if two nodes in a child’s child level (2 levels deeper) are visible, you are on your own. In this case, either rely on GtkTreeStore to emit all signals because it does not implement reference counting, or for models that do implement reference counting, obtain references on these child levels yourself.

Functions

GtkTreeModelFilterVisibleFunc ()

```
gboolean
(*GtkTreeModelFilterVisibleFunc) (GtkTreeModel *model,
                                  GtkTreeIter *iter,
                                  gpointer data);
```

A function which decides whether the row indicated by `iter` is visible.

Parameters

model	the child model of the GtkTreeModelFilter	
iter	a GtkTreeIter pointing to the row in `model` whose visibility is determined	
data	user data given to `gtk_tree_model_filter_set_visible_func()`.	[closure]

Returns

Whether the row indicated by `iter` is visible.

GtkTreeModelFilterModifyFunc ()

```
void
(*GtkTreeModelFilterModifyFunc) (GtkTreeModel *model,
                                 GtkTreeIter *iter,
                                 GValue *value,
                                 gint column,
                                 gpointer data);
```

A function which calculates display values from raw values in the model. It must fill `value` with the display value for the column `column` in the row indicated by `iter` .

Since this function is called for each data access, it's not a particularly efficient operation.

Parameters

model	the GtkTreeModelFilter	
iter	a GtkTreeIter pointing to the row whose display values are determined	
value	A GValue which is already initialized for with the correct type for the column `column` .	[out caller-allocates]
column	the column whose display value is determined	

data	user data given to gtk_tree_model_filter_set_modify_func().	[closure]

---

gtk_tree_model_filter_new ()

```
GtkTreeModel *
gtk_tree_model_filter_new (GtkTreeModel *child_model,
                           GtkTreePath *root);
```

Creates a new GtkTreeModel, with `child_model` as the child_model and `root` as the virtual root.

Parameters

child_model	A GtkTreeModel.	
root	A GtkTreePath or `NULL`.	[allow-none]

Returns

A new GtkTreeModel.

[transfer full]

Since: 2.4

---

gtk_tree_model_filter_set_visible_func ()

```
void
gtk_tree_model_filter_set_visible_func
                                (GtkTreeModelFilter *filter,
                                 GtkTreeModelFilterVisibleFunc func,
                                 gpointer data,
                                 GDestroyNotify destroy);
```

Sets the visible function used when filtering the `filter` to be `func` . The function should return `TRUE` if the given row should be visible and `FALSE` otherwise.

If the condition calculated by the function changes over time (e.g. because it depends on some global parameters), you must call gtk_tree_model_filter_refilter() to keep the visibility information of the model up-to-date.

Note that `func` is called whenever a row is inserted, when it may still be empty. The visible function should therefore take special care of empty rows, like in the example below.

```
1                                       static gboolean
2                                       visible_func (GtkTreeModel *model,
3                                                     GtkTreeIter  *iter,
4                                                     gpointer      data)
5                                       {
6                                         // Visible if row is non-empty and first
7                                       column is “HI”
8                                         gchar *str;
9                                         gboolean visible = FALSE;
10
```

```
11                                          gtk_tree_model_get (model, iter, 0, &str, -
12                                        1);
13                                          if (str && strcmp (str, "HI") == 0)
14                                            visible = TRUE;
15                                          g_free (str);
16
                                            return visible;
                                          }
```

Note that `gtk_tree_model_filter_set_visible_func()` or `gtk_tree_model_filter_set_visible_column()` can only be called once for a given filter model.

Parameters

filter	A GtkTreeModelFilter	
func	A GtkTreeModelFilterVisibleFunc, the visible function	
data	User data to pass to the visible function, or `NULL`.	[allow-none]
destroy	Destroy notifier of `data` , or `NULL`.	[allow-none]

Since: 2.4

---

gtk_tree_model_filter_set_modify_func ()

```
void
gtk_tree_model_filter_set_modify_func (GtkTreeModelFilter *filter,
                                       gint n_columns,
                                       GType *types,
                                       GtkTreeModelFilterModifyFunc func,
                                       gpointer data,
                                       GDestroyNotify destroy);
```

With the `n_columns` and `types` parameters, you give an array of column types for this model (which will be exposed to the parent model/view). The `func` , `data` and `destroy` parameters are for specifying the modify function. The modify function will get called for each data access, the goal of the modify function is to return the data which should be displayed at the location specified using the parameters of the modify function.

Note that `gtk_tree_model_filter_set_modify_func()` can only be called once for a given filter model.

Parameters

filter	A GtkTreeModelFilter.	
n_columns	The number of columns in the filter model.	
types	The GTypes of the columns.	[array length=n_columns]
func	A GtkTreeModelFilterModifyFunc	
data	User data to pass to the modify function, or `NULL`.	[allow-none]
destroy	Destroy notifier of `data` , or `NULL`.	[allow-none]

Since: 2.4

---

gtk_tree_model_filter_set_visible_column ()

```
void
gtk_tree_model_filter_set_visible_column
                                (GtkTreeModelFilter *filter,
                                 gint column);
```

Sets `column` of the child_model to be the column where `filter` should look for visibility information. `columns` should be a column of type `G_TYPE_BOOLEAN`, where `TRUE` means that a row is visible, and `FALSE` if not.

Note that `gtk_tree_model_filter_set_visible_func()` or `gtk_tree_model_filter_set_visible_column()` can only be called once for a given filter model.

Parameters

filter	A GtkTreeModelFilter
column	A gint which is the column containing the visible information

Since: 2.4

---

gtk_tree_model_filter_get_model ()

```
GtkTreeModel *
gtk_tree_model_filter_get_model (GtkTreeModelFilter *filter);
```

Returns a pointer to the child model of `filter` .

Parameters

filter	A GtkTreeModelFilter.

Returns

A pointer to a GtkTreeModel.

[transfer none]

Since: 2.4

---

gtk_tree_model_filter_convert_child_iter_to_iter ()

```
gboolean
gtk_tree_model_filter_convert_child_iter_to_iter
                                (GtkTreeModelFilter *filter,
                                 GtkTreeIter *filter_iter,
                                 GtkTreeIter *child_iter);
```

Sets `filter_iter` to point to the row in `filter` that corresponds to the row pointed at by `child_iter` . If `filter_iter` was not set, `FALSE` is returned.

Parameters

filter	A GtkTreeModelFilter.	
filter_iter	An uninitialized GtkTreeIter.	[out]
child_iter	A valid GtkTreeIter pointing to a row on the child model.	

Returns

`TRUE`, if `filter_iter` was set, i.e. if `child_iter` is a valid iterator pointing to a visible row in child model.

Since: 2.4

---

gtk_tree_model_filter_convert_iter_to_child_iter ()

```
void
gtk_tree_model_filter_convert_iter_to_child_iter
                                (GtkTreeModelFilter *filter,
                                 GtkTreeIter *child_iter,
                                 GtkTreeIter *filter_iter);
```

Sets `child_iter` to point to the row pointed to by `filter_iter` .

Parameters

filter	A GtkTreeModelFilter.	
child_iter	An uninitialized GtkTreeIter.	[out]
filter_iter	A valid GtkTreeIter pointing to a row on `filter` .	

Since: 2.4

---

gtk_tree_model_filter_convert_child_path_to_path ()

```
GtkTreePath *
gtk_tree_model_filter_convert_child_path_to_path
                                (GtkTreeModelFilter *filter,
                                 GtkTreePath *child_path);
```

Converts `child_path` to a path relative to `filter` . That is, `child_path` points to a path in the child model. The rerturned path will point to the same row in the filtered model. If `child_path` isn't a valid path on the child model or points to a row which is not visible in `filter` , then `NULL` is returned.

Parameters

filter	A GtkTreeModelFilter.	
child_path	A GtkTreePath to convert.	

Returns

A newly allocated GtkTreePath, or `NULL`.

[nullable][transfer full]

Since: 2.4

---

gtk_tree_model_filter_convert_path_to_child_path ()

```
GtkTreePath *
gtk_tree_model_filter_convert_path_to_child_path
                                (GtkTreeModelFilter *filter,
                                 GtkTreePath *filter_path);
```

Converts `filter_path` to a path on the child model of `filter` . That is, `filter_path` points to a location in `filter` . The returned path will point to the same location in the model not being filtered. If `filter_path` does not point to a location in the child model, `NULL` is returned.

Parameters

filter	A GtkTreeModelFilter.
filter_path	A GtkTreePath to convert.

Returns

A newly allocated GtkTreePath, or `NULL`.

[nullable][transfer full]

Since: 2.4

---

gtk_tree_model_filter_refilter ()

```
void
gtk_tree_model_filter_refilter (GtkTreeModelFilter *filter);
```

Emits ::row_changed for each row in the child model, which causes the filter to re-evaluate whether a row is visible or not.

Parameters

filter	A GtkTreeModelFilter.

Since: 2.4

---

gtk_tree_model_filter_clear_cache ()

```
void
gtk_tree_model_filter_clear_cache (GtkTreeModelFilter *filter);
```

This function should almost never be called. It clears the `filter` of any cached iterators that haven't been reffed with `gtk_tree_model_ref_node()`. This might be useful if the child model being filtered is static (and doesn't change often) and there has been a lot of unreffed access to nodes. As a side effect of this function, all unreffed iters will be invalid.

Parameters

filter	A GtkTreeModelFilter.

Since: 2.4

Types and Values

struct GtkTreeModelFilter

```
struct GtkTreeModelFilter;
```

Property Details

The "child-model" property

```
  "child-model"              GtkTreeModel *
```

The model for the filtermodel to filter.

Flags: Read / Write / Construct Only

The "virtual-root" property

```
  "virtual-root"             GtkTreePath *
```

The virtual root (relative to the child model) for this filtermodel.

Flags: Read / Write / Construct Only

See Also

GtkTreeModelSort

GtkCellLayout

GtkCellLayout — An interface for packing cells

Functions

void	(*GtkCellLayoutDataFunc) ()
void	gtk_cell_layout_pack_start ()
void	gtk_cell_layout_pack_end ()
GtkCellArea *	gtk_cell_layout_get_area ()
GList *	gtk_cell_layout_get_cells ()
void	gtk_cell_layout_reorder ()
void	gtk_cell_layout_clear ()
void	gtk_cell_layout_set_attributes ()
void	gtk_cell_layout_add_attribute ()
void	gtk_cell_layout_set_cell_data_func ()
void	gtk_cell_layout_clear_attributes ()

Types and Values

	GtkCellLayout
struct	GtkCellLayoutIface

Object Hierarchy

```
GInterface
└── GtkCellLayout
```

Prerequisites

GtkCellLayout requires GObject.

Known Implementations

GtkCellLayout is implemented by GtkAppChooserButton, GtkCellArea, GtkCellAreaBox, GtkCellView, GtkComboBox, GtkComboBoxText, GtkEntryCompletion, GtkIconView and GtkTreeViewColumn.

Includes

```
#include <gtk/gtk.h>
```

Description

GtkCellLayout is an interface to be implemented by all objects which want to provide a GtkTreeViewColumn like API for packing cells, setting attributes and data funcs.

One of the notable features provided by implementations of GtkCellLayout are attributes. Attributes let you set the properties in flexible ways. They can just be set to constant values like regular properties. But they can also be mapped to a column of the underlying tree model with `gtk_cell_layout_set_attributes()`, which means that the value of the attribute can change from cell to cell as they are rendered by the cell renderer. Finally, it is possible to specify a function with `gtk_cell_layout_set_cell_data_func()` that is called to determine the value of the attribute for each cell that is rendered.

GtkCellLayouts as GtkBuildable

Implementations of GtkCellLayout which also implement the GtkBuildable interface (GtkCellView, GtkIconView, GtkComboBox, GtkEntryCompletion, GtkTreeViewColumn) accept GtkCellRenderer objects as <child> elements in UI definitions. They support a custom <attributes> element for their children, which can contain multiple <attribute> elements. Each <attribute> element has a name attribute which specifies a property of the cell renderer; the content of the element is the attribute value.

This is an example of a UI definition fragment specifying attributes:

```
<object class="GtkCellView">
  <child>
    <object class="GtkCellRendererText"/>
    <attributes>
      <attribute name="text">0</attribute>
    </attributes>
  </child>"
</object>
```

Furthermore for implementations of GtkCellLayout that use a GtkCellArea to lay out cells (all GtkCellLayouts in GTK+ use a GtkCellArea) cell properties can also be defined in the format by specifying the custom <cell-packing> attribute which can contain multiple <property> elements defined in the normal way.

Here is a UI definition fragment specifying cell properties:

```
<object class="GtkTreeViewColumn">
  <child>
    <object class="GtkCellRendererText"/>
    <cell-packing>
      <property name="align">True</property>
      <property
name="expand">False</property>
    </cell-packing>
  </child>"
</object>
```

Subclassing GtkCellLayout implementations

When subclassing a widget that implements GtkCellLayout like GtkIconView or GtkComboBox, there are some considerations related to the fact that these widgets internally use a GtkCellArea. The cell area is exposed as a construct-only property by these widgets. This means that it is possible to e.g. do

```
combo = g_object_new (GTK_TYPE_COMBO_BOX,
"cell-area", my_cell_area, NULL);
```

to use a custom cell area with a combo box. But construct properties are only initialized after instance `init()` functions have run, which means that using functions which rely on the existence of the cell area in your subclass' `init()` function will cause the default cell area to be instantiated. In this case, a provided construct property value will be ignored (with a warning, to alert you to the problem).

```
static void
my_combo_box_init (MyComboBox *b)
{
  GtkCellRenderer *cell;

  cell = gtk_cell_renderer_pixbuf_new ();
  // The following call causes the default
cell area for combo boxes,
  // a GtkCellAreaBox, to be instantiated
  gtk_cell_layout_pack_start (GTK_CELL_LAYOUT
(b), cell, FALSE);
  ...
}
```

```
14
15                                                  GtkWidget *
16                                                  my_combo_box_new (GtkCellArea *area)
17                                                  {
18                                                    // This call is going to cause a warning
                                                    about area being ignored
                                                    return g_object_new (MY_TYPE_COMBO_BOX,
                                                    "cell-area", area, NULL);
                                                    }
```

If supporting alternative cell areas with your derived widget is not important, then this does not have to concern you. If you want to support alternative cell areas, you can do so by moving the problematic calls out of `init()` and into a `constructor()` for your class.

Functions

GtkCellLayoutDataFunc ()

```
void
(*GtkCellLayoutDataFunc) (GtkCellLayout *cell_layout,
                          GtkCellRenderer *cell,
                          GtkTreeModel *tree_model,
                          GtkTreeIter *iter,
                          gpointer data);
```

A function which should set the value of `cell_layout` 's cell renderer(s) as appropriate.

Parameters

cell_layout	a GtkCellLayout	
cell	the cell renderer whose value is to be set	
tree_model	the model	
iter	a GtkTreeIter indicating the row to set the value for	
data	user data passed to `gtk_cell_layout_set_cell_data_func()`.	[closure]

---

gtk_cell_layout_pack_start ()

```
void
gtk_cell_layout_pack_start (GtkCellLayout *cell_layout,
                            GtkCellRenderer *cell,
                            gboolean expand);
```

Packs the `cell` into the beginning of `cell_layout` . If `expand` is `FALSE`, then the `cell` is allocated no more space than it needs. Any unused space is divided evenly between cells for which `expand` is `TRUE`.

Note that reusing the same cell renderer is not supported.

Parameters

cell_layout	a GtkCellLayout
cell	a GtkCellRenderer
expand	TRUE if cell is to be given extra space allocated to cell_layout

Since: 2.4

---

gtk_cell_layout_pack_end ()

```
void
gtk_cell_layout_pack_end (GtkCellLayout *cell_layout,
                          GtkCellRenderer *cell,
                          gboolean expand);
```

Adds the cell to the end of cell_layout . If expand is FALSE, then the cell is allocated no more space than it needs. Any unused space is divided evenly between cells for which expand is TRUE.

Note that reusing the same cell renderer is not supported.

Parameters

cell_layout	a GtkCellLayout
cell	a GtkCellRenderer
expand	TRUE if cell is to be given extra space allocated to cell_layout

Since: 2.4

---

gtk_cell_layout_get_area ()

```
GtkCellArea *
gtk_cell_layout_get_area (GtkCellLayout *cell_layout);
```

Returns the underlying GtkCellArea which might be cell_layout if called on a GtkCellArea or might be NULL if no GtkCellArea is used by cell_layout .

Parameters

cell_layout	a GtkCellLayout

Returns

the cell area used by cell_layout , or NULL in case no cell area is used.

[transfer none][nullable]

Since: 3.0

---

gtk_cell_layout_get_cells ()

```
GList *
gtk_cell_layout_get_cells (GtkCellLayout *cell_layout);
```

Returns the cell renderers which have been added to `cell_layout` .

Parameters

cell_layout	a GtkCellLayout

Returns

a list of cell renderers. The list, but not the renderers has been newly allocated and should be freed with `g_list_free()` when no longer needed.

[element-type GtkCellRenderer][transfer container]

Since: 2.12

---

gtk_cell_layout_reorder ()

```
void
gtk_cell_layout_reorder (GtkCellLayout *cell_layout,
                         GtkCellRenderer *cell,
                         gint position);
```

Re-inserts `cell` at `position` .

Note that `cell` has already to be packed into `cell_layout` for this to function properly.

Parameters

cell_layout	a GtkCellLayout
cell	a GtkCellRenderer to reorder
position	new position to insert `cell` at

Since: 2.4

---

gtk_cell_layout_clear ()

```
void
gtk_cell_layout_clear (GtkCellLayout *cell_layout);
```

Unsets all the mappings on all renderers on `cell_layout` and removes all renderers from `cell_layout` .

Parameters

cell_layout	a GtkCellLayout

Since: 2.4

---

gtk_cell_layout_set_attributes ()

```
void
gtk_cell_layout_set_attributes (GtkCellLayout *cell_layout,
                                GtkCellRenderer *cell,
                                ...);
```

Sets the attributes in list as the attributes of `cell_layout` .

The attributes should be in attribute/column order, as in `gtk_cell_layout_add_attribute()`. All existing attributes are removed, and replaced with the new attributes.

Parameters

cell_layout	a GtkCellLayout
cell	a GtkCellRenderer
...	a `NULL`-terminated list of attributes

Since: 2.4

---

gtk_cell_layout_add_attribute ()

```
void
gtk_cell_layout_add_attribute (GtkCellLayout *cell_layout,
                               GtkCellRenderer *cell,
                               const gchar *attribute,
                               gint column);
```

Adds an attribute mapping to the list in `cell_layout` .

The `column` is the column of the model to get a value from, and the `attribute` is the parameter on `cell` to be set from the value. So for example if column 2 of the model contains strings, you could have the “text” attribute of a GtkCellRendererText get its values from column 2.

Parameters

cell_layout	a GtkCellLayout
cell	a GtkCellRenderer
attribute	an attribute on the renderer
column	the column position on the model to get the attribute from

Since: 2.4

---

gtk_cell_layout_set_cell_data_func ()

```
void
gtk_cell_layout_set_cell_data_func (GtkCellLayout *cell_layout,
                                    GtkCellRenderer *cell,
                                    GtkCellLayoutDataFunc func,
                                    gpointer func_data,
```

```
                                    GDestroyNotify destroy);
```

Sets the GtkCellLayoutDataFunc to use for `cell_layout` .

This function is used instead of the standard attributes mapping for setting the column value, and should set the value of `cell_layout` 's cell renderer(s) as appropriate.

`func` may be `NULL` to remove a previously set function.

Parameters

cell_layout	a GtkCellLayout	
cell	a GtkCellRenderer	
func	the GtkCellLayoutDataFunc to use, or `NULL`.	[allow-none]
func_data	user data for `func` .	[closure]
destroy	destroy notify for `func_data`	

Since: 2.4

---

gtk_cell_layout_clear_attributes ()

```
void
gtk_cell_layout_clear_attributes (GtkCellLayout *cell_layout,
                                  GtkCellRenderer *cell);
```

Clears all existing attributes previously set with `gtk_cell_layout_set_attributes()`.

Parameters

cell_layout	a GtkCellLayout
cell	a GtkCellRenderer to clear the attribute mapping on

Since: 2.4

Types and Values

GtkCellLayout

```
typedef struct _GtkCellLayout GtkCellLayout;
```

---

struct GtkCellLayoutIface

```
struct GtkCellLayoutIface {
  /* Virtual Table */
  void (* pack_start)         (GtkCellLayout         *cell_layout,
                               GtkCellRenderer       *cell,
                               gboolean               expand);
  void (* pack_end)           (GtkCellLayout         *cell_layout,
```

```
                                GtkCellRenderer       *cell,
                                gboolean               expand);
  void (* clear)               (GtkCellLayout         *cell_layout);
  void (* add_attribute)       (GtkCellLayout         *cell_layout,
                                GtkCellRenderer       *cell,
                                const gchar           *attribute,
                                gint                   column);
  void (* set_cell_data_func) (GtkCellLayout         *cell_layout,
                                GtkCellRenderer       *cell,
                                GtkCellLayoutDataFunc  func,
                                gpointer               func_data,
                                GDestroyNotify         destroy);
  void (* clear_attributes)    (GtkCellLayout         *cell_layout,
                                GtkCellRenderer       *cell);
  void (* reorder)             (GtkCellLayout         *cell_layout,
                                GtkCellRenderer       *cell,
                                gint                   position);
  GList* (* get_cells)         (GtkCellLayout         *cell_layout);

  GtkCellArea *(* get_area)    (GtkCellLayout         *cell_layout);
};
```

Members

`pack_start` ()	Packs the cell into the beginning of cell_layout.
`pack_end` ()	Adds the cell to the end of cell_layout.
`clear` ()	Unsets all the mappings on all renderers on cell_layout and removes all renderers from cell_layout.
`add_attribute` ()	Adds an attribute mapping to the list in cell_layout.
`set_cell_data_func` ()	Sets the GtkCellLayoutDataFunc to use for cell_layout.
`clear_attributes` ()	Clears all existing attributes previously set with `gtk_cell_layout_set_attributes()`.
`reorder` ()	Re-inserts cell at position.
`get_cells` ()	Get the cell renderers which have been added to cell_layout.
`get_area` ()	Get the underlying GtkCellArea which might be cell_layout if called on a GtkCellArea or might be NULL if no GtkCellArea is used by cell_layout.

GtkCellArea

GtkCellArea — An abstract class for laying out GtkCellRenderers

Functions

gboolean	(*GtkCellCallback) ()
gboolean	(*GtkCellAllocCallback) ()
#define	GTK_CELL_AREA_WARN_INVALID_CELL_PROPERTY_ID()
void	gtk_cell_area_add ()
void	gtk_cell_area_remove ()
gboolean	gtk_cell_area_has_renderer ()
void	gtk_cell_area_foreach ()
void	gtk_cell_area_foreach_alloc ()
gint	gtk_cell_area_event ()
void	gtk_cell_area_render ()
void	gtk_cell_area_get_cell_allocation ()
GtkCellRenderer *	gtk_cell_area_get_cell_at_position ()
GtkCellAreaContext *	gtk_cell_area_create_context ()
GtkCellAreaContext *	gtk_cell_area_copy_context ()
GtkSizeRequestMode	gtk_cell_area_get_request_mode ()
void	gtk_cell_area_get_preferred_width ()
void	gtk_cell_area_get_preferred_height_for_width ()
void	gtk_cell_area_get_preferred_height ()
void	gtk_cell_area_get_preferred_width_for_height ()
const gchar *	gtk_cell_area_get_current_path_string ()
void	gtk_cell_area_apply_attributes ()
void	gtk_cell_area_attribute_connect ()
void	gtk_cell_area_attribute_disconnect ()
gint	gtk_cell_area_attribute_get_column ()
void	gtk_cell_area_class_install_cell_property ()
GParamSpec *	gtk_cell_area_class_find_cell_property ()
GParamSpec **	gtk_cell_area_class_list_cell_properties ()
void	gtk_cell_area_add_with_properties ()
void	gtk_cell_area_cell_set ()
void	gtk_cell_area_cell_get ()
void	gtk_cell_area_cell_set_valist ()
void	gtk_cell_area_cell_get_valist ()
void	gtk_cell_area_cell_set_property ()
void	gtk_cell_area_cell_get_property ()
gboolean	gtk_cell_area_is_activatable ()
gboolean	gtk_cell_area_activate ()
gboolean	gtk_cell_area_focus ()
void	gtk_cell_area_set_focus_cell ()
GtkCellRenderer *	gtk_cell_area_get_focus_cell ()
void	gtk_cell_area_add_focus_sibling ()
void	gtk_cell_area_remove_focus_sibling ()
gboolean	gtk_cell_area_is_focus_sibling ()
const GList *	gtk_cell_area_get_focus_siblings ()
GtkCellRenderer *	gtk_cell_area_get_focus_from_sibling ()
GtkCellRenderer *	gtk_cell_area_get_edited_cell ()
GtkCellEditable *	gtk_cell_area_get_edit_widget ()
gboolean	gtk_cell_area_activate_cell ()

void	gtk_cell_area_stop_editing ()
void	gtk_cell_area_inner_cell_area ()
void	gtk_cell_area_request_renderer ()

Properties

GtkCellEditable *	edit-widget	Read
GtkCellRenderer *	edited-cell	Read
GtkCellRenderer *	focus-cell	Read / Write

Signals

void	add-editable	Run First
void	apply-attributes	Run First
void	focus-changed	Run First
void	remove-editable	Run First

Types and Values

struct	GtkCellArea
struct	GtkCellAreaClass

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkCellArea
        └── GtkCellAreaBox
```

Implemented Interfaces

GtkCellArea implements GtkCellLayout and GtkBuildable.

Includes

```
#include <gtk/gtk.h>
```

Description

The GtkCellArea is an abstract class for GtkCellLayout widgets (also referred to as "layouting widgets") to interface with an arbitrary number of GtkCellRenderers and interact with the user for a given GtkTreeModel row.

The cell area handles events, focus navigation, drawing and size requests and allocations for a given row of data.

Usually users dont have to interact with the GtkCellArea directly unless they are implementing a cell-layouting widget themselves.

Requesting area sizes

As outlined in GtkWidget's geometry management section, GTK+ uses a height-for-width geometry management system to compute the sizes of widgets and user interfaces. GtkCellArea uses the same semantics to calculate the size of an area for an arbitrary number of GtkTreeModel rows.

When requesting the size of a cell area one needs to calculate the size for a handful of rows, and this will be done differently by different layouting widgets. For instance a GtkTreeViewColumn always lines up the areas from top to bottom while a GtkIconView on the other hand might enforce that all areas received the same width and wrap the areas around, requesting height for more cell areas when allocated less width.

It's also important for areas to maintain some cell alignments with areas rendered for adjacent rows (cells can appear "columnized" inside an area even when the size of cells are different in each row). For this reason the GtkCellArea uses a GtkCellAreaContext object to store the alignments and sizes along the way (as well as the overall largest minimum and natural size for all the rows which have been calculated with the said context).

The GtkCellAreaContext is an opaque object specific to the GtkCellArea which created it (see `gtk_cell_area_create_context()`). The owning cell-layouting widget can create as many contexts as it wishes to calculate sizes of rows which should receive the same size in at least one orientation (horizontally or vertically), However, it's important that the same GtkCellAreaContext which was used to request the sizes for a given GtkTreeModel row be used when rendering or processing events for that row.

In order to request the width of all the rows at the root level of a GtkTreeModel one would do the following:

```
GtkTreeIter iter;
gint        minimum_width;
gint        natural_width;

valid = gtk_tree_model_get_iter_first (model,
&iter);
while (valid)
  {
    gtk_cell_area_apply_attributes (area,
model, &iter, FALSE, FALSE);
    gtk_cell_area_get_preferred_width (area,
context, widget, NULL, NULL);

    valid = gtk_tree_model_iter_next (model,
&iter);
  }
gtk_cell_area_context_get_preferred_width
(context, &minimum_width, &natural_width);
```

Note that in this example it's not important to observe the returned minimum and natural width of the area for each row unless the cell-layouting object is actually interested in the widths of individual rows. The overall width is however stored in the accompanying GtkCellAreaContext object and can be consulted at any time.

This can be useful since GtkCellLayout widgets usually have to support requesting and rendering rows in treemodels with an exceedingly large amount of rows. The GtkCellLayout widget in that case would calculate the required width of the rows in an idle or timeout source (see `g_timeout_add()`) and when the widget is requested its actual width in `GtkWidgetClass.get_preferred_width()` it can simply consult the width accumulated so far in the GtkCellAreaContext object.

A simple example where rows are rendered from top to bottom and take up the full width of the layouting widget would look like:

```
static void
foo_get_preferred_width (GtkWidget
*widget,
                         gint
*minimum_size,
                         gint
```

```
7                                                   *natural_size)
8                                                   {
9                                                     Foo        *foo  = FOO (widget);
10                                                    FooPrivate *priv = foo->priv;
11
12

                                                    foo_ensure_at_least_one_handfull_of_rows_have
                                                    _been_requested (foo);

                                                      gtk_cell_area_context_get_preferred_width
                                                    (priv->context, minimum_size, natural_size);
                                                    }
```

In the above example the Foo widget has to make sure that some row sizes have been calculated (the amount of rows that Foo judged was appropriate to request space for in a single timeout iteration) before simply returning the amount of space required by the area via the GtkCellAreaContext.

Requesting the height for width (or width for height) of an area is a similar task except in this case the GtkCellAreaContext does not store the data (actually, it does not know how much space the layouting widget plans to allocate it for every row. It's up to the layouting widget to render each row of data with the appropriate height and width which was requested by the GtkCellArea).

In order to request the height for width of all the rows at the root level of a GtkTreeModel one would do the following:

```
1                                                   GtkTreeIter iter;
2                                                   gint        minimum_height;
3                                                   gint        natural_height;
4                                                   gint        full_minimum_height = 0;
5                                                   gint        full_natural_height = 0;
6
7                                                   valid = gtk_tree_model_get_iter_first (model,
8                                                   &iter);
9                                                   while (valid)
10                                                    {
11                                                    gtk_cell_area_apply_attributes (area,
12                                                  model, &iter, FALSE, FALSE);
13
14                                                  gtk_cell_area_get_preferred_height_for_width
15                                                  (area, context, widget,
16
17                                                  width, &minimum_height, &natural_height);
18
19                                                      if (width_is_for_allocation)
20                                                         cache_row_height (&iter,
21                                                  minimum_height, natural_height);

                                                        full_minimum_height += minimum_height;
                                                        full_natural_height += natural_height;

                                                        valid = gtk_tree_model_iter_next (model,
                                                  &iter);
                                                    }
```

Note that in the above example we would need to cache the heights returned for each row so that we would know what sizes to render the areas for each row. However we would only want to really cache the heights if the request is intended for the layouting widgets real allocation.

In some cases the layouting widget is requested the height for an arbitrary for_width, this is a special case for layouting widgets who need to request size for tens of thousands of rows. For this case it's only important that the layouting widget calculate one reasonably sized chunk of rows and return that height synchronously. The reasoning here is that any layouting widget is at least capable of synchronously calculating enough height to fill the screen height (or scrolled window height) in response to a single call to

GtkWidgetClass.get_preferred_height_for_width(). Returning a perfect height for width that is larger than the screen area is inconsequential since after the layouting receives an allocation from a scrolled window it simply continues to drive the scrollbar values while more and more height is required for the row heights that are calculated in the background.

Rendering Areas

Once area sizes have been aquired at least for the rows in the visible area of the layouting widget they can be rendered at GtkWidgetClass.draw() time.

A crude example of how to render all the rows at the root level runs as follows:

```
GtkAllocation allocation;
GdkRectangle  cell_area = { 0, };
GtkTreeIter   iter;
gint          minimum_width;
gint          natural_width;

gtk_widget_get_allocation (widget,
&allocation);
cell_area.width = allocation.width;

valid = gtk_tree_model_get_iter_first (model,
&iter);
while (valid)
  {
    cell_area.height =
get_cached_height_for_row (&iter);

    gtk_cell_area_apply_attributes (area,
model, &iter, FALSE, FALSE);
    gtk_cell_area_render (area, context,
widget, cr,
                          &cell_area,
&cell_area, state_flags, FALSE);

    cell_area.y += cell_area.height;

    valid = gtk_tree_model_iter_next (model,
&iter);
  }
```

Note that the cached height in this example really depends on how the layouting widget works. The layouting widget might decide to give every row its minimum or natural height or, if the model content is expected to fit inside the layouting widget without scrolling, it would make sense to calculate the allocation for each row at "size-allocate" time using gtk_distribute_natural_allocation().

Handling Events and Driving Keyboard Focus

Passing events to the area is as simple as handling events on any normal widget and then passing them to the gtk_cell_area_event() API as they come in. Usually GtkCellArea is only interested in button events, however some customized derived areas can be implemented who are interested in handling other events. Handling an event can trigger the "focus-changed" signal to fire; as well as "add-editable" in the case that an editable cell was clicked and needs to start editing. You can call gtk_cell_area_stop_editing() at any time to cancel any cell editing that is currently in progress.

The GtkCellArea drives keyboard focus from cell to cell in a way similar to GtkWidget. For layouting widgets that support giving focus to cells it's important to remember to pass `GTK_CELL_RENDERER_FOCUSED` to the area functions for the row that has focus and to tell the area to paint the focus at render time.

Layouting widgets that accept focus on cells should implement the `GtkWidgetClass.focus()` virtual method. The layouting widget is always responsible for knowing where GtkTreeModel rows are rendered inside the widget, so at `GtkWidgetClass.focus()` time the layouting widget should use the GtkCellArea methods to navigate focus inside the area and then observe the GtkDirectionType to pass the focus to adjacent rows and areas.

A basic example of how the `GtkWidgetClass.focus()` virtual method should be implemented:

```
static gboolean
foo_focus (GtkWidget       *widget,
           GtkDirectionType direction)
{
  Foo        *foo  = FOO (widget);
  FooPrivate *priv = foo->priv;
  gint        focus_row;
  gboolean    have_focus = FALSE;

  focus_row = priv->focus_row;

  if (!gtk_widget_has_focus (widget))
    gtk_widget_grab_focus (widget);

  valid = gtk_tree_model_iter_nth_child
(priv->model, &iter, NULL, priv->focus_row);
  while (valid)
    {
      gtk_cell_area_apply_attributes (priv-
>area, priv->model, &iter, FALSE, FALSE);

      if (gtk_cell_area_focus (priv->area,
direction))
        {
           priv->focus_row = focus_row;
           have_focus = TRUE;
           break;
        }
      else
        {
          if (direction == GTK_DIR_RIGHT ||
              direction == GTK_DIR_LEFT)
            break;
          else if (direction == GTK_DIR_UP ||
                   direction ==
GTK_DIR_TAB_BACKWARD)
           {
              if (focus_row == 0)
                break;
              else
               {
                  focus_row--;
                  valid =
gtk_tree_model_iter_nth_child (priv->model,
&iter, NULL, focus_row);
               }
           }
          else
            {
              if (focus_row == last_row)
```

```
51                                                      break;
52                                                    else
53                                                      {
54                                                        focus_row++;
55                                                        valid =
                                                gtk_tree_model_iter_next (priv->model,
                                                &iter);
                                                      }
                                                  }
                                              }
                                          }
                                          return have_focus;
                                      }
```

Note that the layouting widget is responsible for matching the GtkDirectionType values to the way it lays out its cells.

Cell Properties

The GtkCellArea introduces cell properties for GtkCellRenderers in very much the same way that GtkContainer introduces child properties for GtkWidgets. This provides some general interfaces for defining the relationship cell areas have with their cells. For instance in a GtkCellAreaBox a cell might “expand” and receive extra space when the area is allocated more than its full natural request, or a cell might be configured to “align” with adjacent rows which were requested and rendered with the same GtkCellAreaContext.

Use `gtk_cell_area_class_install_cell_property()` to install cell properties for a cell area class and `gtk_cell_area_class_find_cell_property()` or `gtk_cell_area_class_list_cell_properties()` to get information about existing cell properties.

To set the value of a cell property, use `gtk_cell_area_cell_set_property()`, `gtk_cell_area_cell_set()` or `gtk_cell_area_cell_set_valist()`. To obtain the value of a cell property, use `gtk_cell_area_cell_get_property()`, `gtk_cell_area_cell_get()` or `gtk_cell_area_cell_get_valist()`.

Functions

GtkCellCallback ()

```
gboolean
(*GtkCellCallback) (GtkCellRenderer *renderer,
                    gpointer data);
```

The type of the callback functions used for iterating over the cell renderers of a GtkCellArea, see `gtk_cell_area_foreach()`.

Parameters

renderer	the cell renderer to operate on	
data	user-supplied data.	[closure]

Returns

TRUE to stop iterating over cells.

GtkCellAllocCallback ()

```
gboolean
(*GtkCellAllocCallback) (GtkCellRenderer *renderer,
                         const GdkRectangle *cell_area,
                         const GdkRectangle *cell_background,
                         gpointer data);
```

The type of the callback functions used for iterating over the cell renderers and their allocated areas inside a GtkCellArea, see gtk_cell_area_foreach_alloc().

Parameters

renderer	the cell renderer to operate on	
cell_area	the area allocated to renderer inside the rectangle provided to gtk_cell_area_foreach_alloc().	
cell_background	the background area for renderer inside the background area provided to gtk_cell_area_foreach_alloc().	
data	user-supplied data.	[closure]

Returns

TRUE to stop iterating over cells.

GTK_CELL_AREA_WARN_INVALID_CELL_PROPERTY_ID()

```
#define GTK_CELL_AREA_WARN_INVALID_CELL_PROPERTY_ID(object, property_id,
pspec)
```

This macro should be used to emit a standard warning about unexpected properties in set_cell_property() and get_cell_property() implementations.

Parameters

object	the GObject on which set_cell_property() or get_cell_property() was called
property_id	the numeric id of the property
pspec	the GParamSpec of the property

---

gtk_cell_area_add ()

```
void
gtk_cell_area_add (GtkCellArea *area,
                   GtkCellRenderer *renderer);
```

Adds `renderer` to `area` with the default child cell properties.

Parameters

area	a GtkCellArea
renderer	the GtkCellRenderer to add to `area`

Since: 3.0

---

gtk_cell_area_remove ()

```
void
gtk_cell_area_remove (GtkCellArea *area,
                      GtkCellRenderer *renderer);
```

Removes `renderer` from `area` .

Parameters

area	a GtkCellArea
renderer	the GtkCellRenderer to remove from `area`

Since: 3.0

---

gtk_cell_area_has_renderer ()

```
gboolean
gtk_cell_area_has_renderer (GtkCellArea *area,
                            GtkCellRenderer *renderer);
```

Checks if `area` contains `renderer` .

Parameters

area	a GtkCellArea
renderer	the GtkCellRenderer to check

Returns

`TRUE` if `renderer` is in the `area` .

Since: 3.0

---

gtk_cell_area_foreach ()

```
void
gtk_cell_area_foreach (GtkCellArea *area,
                       GtkCellCallback callback,
                       gpointer callback_data);
```

Calls `callback` for every GtkCellRenderer in `area` .

Parameters

area	a GtkCellArea	
callback	the GtkCellCallback to call.	[scope call]
callback_data	user provided data pointer	

Since: 3.0

---

gtk_cell_area_foreach_alloc ()

```
void
gtk_cell_area_foreach_alloc (GtkCellArea *area,
                             GtkCellAreaContext *context,
                             GtkWidget *widget,
                             const GdkRectangle *cell_area,
                             const GdkRectangle *background_area,
                             GtkCellAllocCallback callback,
                             gpointer callback_data);
```

Calls `callback` for every GtkCellRenderer in `area` with the allocated rectangle inside `cell_area` .

Parameters

area	a GtkCellArea	
context	the GtkCellAreaContext for this row of data.	
widget	the GtkWidget that `area` is rendering to	
cell_area	the `widget` relative coordinates and size for `area`	
background_area	the `widget` relative coordinates of the background area	
callback	the GtkCellAllocCallback to call.	[scope call]
callback_data	user provided data pointer	

Since: 3.0

---

gtk_cell_area_event ()

```
gint
gtk_cell_area_event (GtkCellArea *area,
                     GtkCellAreaContext *context,
```

```
                        GtkWidget *widget,
                        GdkEvent *event,
                        const GdkRectangle *cell_area,
                        GtkCellRendererState flags);
```

Delegates event handling to a GtkCellArea.

Parameters

area	a GtkCellArea
context	the GtkCellAreaContext for this row of data.
widget	the GtkWidget that `area` is rendering to
event	the GdkEvent to handle
cell_area	the `widget` relative coordinates for `area`
flags	the GtkCellRendererState for `area` in this row.

Returns

`TRUE` if the event was handled by `area` .

Since: 3.0

---

gtk_cell_area_render ()

```
void
gtk_cell_area_render (GtkCellArea *area,
                      GtkCellAreaContext *context,
                      GtkWidget *widget,
                      cairo_t *cr,
                      const GdkRectangle *background_area,
                      const GdkRectangle *cell_area,
                      GtkCellRendererState flags,
                      gboolean paint_focus);
```

Renders `area` ’s cells according to `area` ’s layout onto `widget` at the given coordinates.

Parameters

area	a GtkCellArea
context	the GtkCellAreaContext for this row of data.
widget	the GtkWidget that `area` is rendering to
cr	the cairo_t to render with
background_area	the `widget` relative coordinates for `area` ’s background
cell_area	the `widget` relative coordinates for `area`

flags	the GtkCellRendererState for `area` in this row.	
paint_focus	whether `area` should paint focus on focused cells for focused rows or not.	

Since: 3.0

---

gtk_cell_area_get_cell_allocation ()

```
void
gtk_cell_area_get_cell_allocation (GtkCellArea *area,
                                   GtkCellAreaContext *context,
                                   GtkWidget *widget,
                                   GtkCellRenderer *renderer,
                                   const GdkRectangle *cell_area,
                                   GdkRectangle *allocation);
```

Derives the allocation of `renderer` inside `area` if `area` were to be renderered in `cell_area` .

Parameters

area	a GtkCellArea	
context	the GtkCellAreaContext used to hold sizes for `area` .	
widget	the GtkWidget that `area` is rendering on	
renderer	the GtkCellRenderer to get the allocation for	
cell_area	the whole allocated area for `area` in `widget` for this row	
allocation	where to store the allocation for `renderer` .	[out]

Since: 3.0

---

gtk_cell_area_get_cell_at_position ()

```
GtkCellRenderer *
gtk_cell_area_get_cell_at_position (GtkCellArea *area,
                                    GtkCellAreaContext *context,
                                    GtkWidget *widget,
                                    const GdkRectangle *cell_area,
                                    gint x,
                                    gint y,
                                    GdkRectangle *alloc_area);
```

Gets the GtkCellRenderer at `x` and `y` coordinates inside `area` and optionally returns the full cell allocation for it inside `cell_area` .

Parameters

area	a GtkCellArea	
context	the GtkCellAreaContext used to hold sizes for `area` .	
widget	the GtkWidget that `area` is rendering on	
cell_area	the whole allocated area for `area` in `widget` for this row	
x	the x position	
y	the y position	
alloc_area	where to store the inner allocated area of the returned cell renderer, or `NULL`.	[out][allow-none]

Returns

the GtkCellRenderer at `x` and `y` .

[transfer none]

Since: 3.0

---

gtk_cell_area_create_context ()

```
GtkCellAreaContext *
gtk_cell_area_create_context (GtkCellArea *area);
```

Creates a GtkCellAreaContext to be used with `area` for all purposes. GtkCellAreaContext stores geometry information for rows for which it was operated on, it is important to use the same context for the same row of data at all times (i.e. one should render and handle events with the same GtkCellAreaContext which was used to request the size of those rows of data).

Parameters

area	a GtkCellArea

Returns

a newly created GtkCellAreaContext which can be used with `area` .

[transfer full]

Since: 3.0

---

gtk_cell_area_copy_context ()

```
GtkCellAreaContext *
gtk_cell_area_copy_context (GtkCellArea *area,
                            GtkCellAreaContext *context);
```

This is sometimes needed for cases where rows need to share alignments in one orientation but may be separately grouped in the opposing orientation.

For instance, GtkIconView creates all icons (rows) to have the same width and the cells theirin to have the same horizontal alignments. However each row of icons may have a separate collective height. GtkIconView uses this to request the heights of each row based on a context which was already used to request all the row widths that are to be displayed.

Parameters

area	a GtkCellArea
context	the GtkCellAreaContext to copy

Returns

a newly created GtkCellAreaContext copy of `context` .

[transfer full]

Since: 3.0

---

gtk_cell_area_get_request_mode ()

```
GtkSizeRequestMode
gtk_cell_area_get_request_mode (GtkCellArea *area);
```

Gets whether the area prefers a height-for-width layout or a width-for-height layout.

Parameters

area	a GtkCellArea

Returns

The GtkSizeRequestMode preferred by `area` .

Since: 3.0

---

gtk_cell_area_get_preferred_width ()

```
void
gtk_cell_area_get_preferred_width (GtkCellArea *area,
                                   GtkCellAreaContext *context,
                                   GtkWidget *widget,
                                   gint *minimum_width,
                                   gint *natural_width);
```

Retrieves a cell area’s initial minimum and natural width.

`area` will store some geometrical information in `context` along the way; when requesting sizes over an arbitrary number of rows, it’s not important to check the `minimum_width` and `natural_width` of this call but

rather to consult gtk_cell_area_context_get_preferred_width() after a series of requests.

Parameters

area	a GtkCellArea	
context	the GtkCellAreaContext to perform this request with	
widget	the GtkWidget where area will be rendering	
minimum_width	location to store the minimum width, or NULL.	[out][allow-none]
natural_width	location to store the natural width, or NULL.	[out][allow-none]

Since: 3.0

---

gtk_cell_area_get_preferred_height_for_width ()

```
void
gtk_cell_area_get_preferred_height_for_width
                                (GtkCellArea *area,
                                 GtkCellAreaContext *context,
                                 GtkWidget *widget,
                                 gint width,
                                 gint *minimum_height,
                                 gint *natural_height);
```

Retrieves a cell area's minimum and natural height if it would be given the specified width .

area stores some geometrical information in context along the way while calling gtk_cell_area_get_preferred_width(). It's important to perform a series of gtk_cell_area_get_preferred_width() requests with context first and then call gtk_cell_area_get_preferred_height_for_width() on each cell area individually to get the height for width of each fully requested row.

If at some point, the width of a single row changes, it should be requested with gtk_cell_area_get_preferred_width() again and then the full width of the requested rows checked again with gtk_cell_area_context_get_preferred_width().

Parameters

area	a GtkCellArea	
context	the GtkCellAreaContext which has already been requested for widths.	
widget	the GtkWidget where area will be rendering	
width	the width for which to check the height of this area	
minimum_height	location to store the minimum height, or NULL.	[out][allow-none]
natural_height	location to store the natural height,	[out][allow-none]

or NULL.

Since: 3.0

---

gtk_cell_area_get_preferred_height ()

```
void
gtk_cell_area_get_preferred_height (GtkCellArea *area,
                                    GtkCellAreaContext *context,
                                    GtkWidget *widget,
                                    gint *minimum_height,
                                    gint *natural_height);
```

Retrieves a cell area's initial minimum and natural height.

area will store some geometrical information in context along the way; when requesting sizes over an arbitrary number of rows, it's not important to check the minimum_height and natural_height of this call but rather to consult gtk_cell_area_context_get_preferred_height() after a series of requests.

Parameters

area	a GtkCellArea	
context	the GtkCellAreaContext to perform this request with	
widget	the GtkWidget where area will be rendering	
minimum_height	location to store the minimum height, or NULL.	[out][allow-none]
natural_height	location to store the natural height, or NULL.	[out][allow-none]

Since: 3.0

---

gtk_cell_area_get_preferred_width_for_height ()

```
void
gtk_cell_area_get_preferred_width_for_height
                                (GtkCellArea *area,
                                 GtkCellAreaContext *context,
                                 GtkWidget *widget,
                                 gint height,
                                 gint *minimum_width,
                                 gint *natural_width);
```

Retrieves a cell area's minimum and natural width if it would be given the specified height .

area stores some geometrical information in context along the way while calling gtk_cell_area_get_preferred_height(). It's important to perform a series of gtk_cell_area_get_preferred_height() requests with context first and then call gtk_cell_area_get_preferred_width_for_height() on each cell area individually to get the height for width of each fully requested row.

If at some point, the height of a single row changes, it should be requested with gtk_cell_area_get_preferred_height() again and then the full height of the requested rows checked again

with gtk_cell_area_context_get_preferred_height().

Parameters

area	a GtkCellArea	
context	the GtkCellAreaContext which has already been requested for widths.	
widget	the GtkWidget where area will be rendering	
height	the height for which to check the width of this area	
minimum_width	location to store the minimum width, or NULL.	[out][allow-none]
natural_width	location to store the natural width, or NULL.	[out][allow-none]

Since: 3.0

---

gtk_cell_area_get_current_path_string ()

```
const gchar *
gtk_cell_area_get_current_path_string (GtkCellArea *area);
```

Gets the current GtkTreePath string for the currently applied GtkTreeIter, this is implicitly updated when gtk_cell_area_apply_attributes() is called and can be used to interact with renderers from GtkCellArea subclasses.

Parameters

area	a GtkCellArea

Returns

The current GtkTreePath string for the current attributes applied to area . This string belongs to the area and should not be freed.

Since: 3.0

---

gtk_cell_area_apply_attributes ()

```
void
gtk_cell_area_apply_attributes (GtkCellArea *area,
                                GtkTreeModel *tree_model,
                                GtkTreeIter *iter,
                                gboolean is_expander,
                                gboolean is_expanded);
```

Applies any connected attributes to the renderers in area by pulling the values from tree_model .

Parameters

area	a GtkCellArea
tree_model	the GtkTreeModel to pull values from
iter	the GtkTreeIter in `tree_model` to apply values for
is_expander	whether `iter` has children
is_expanded	whether `iter` is expanded in the view and children are visible

Since: 3.0

---

gtk_cell_area_attribute_connect ()

```
void
gtk_cell_area_attribute_connect (GtkCellArea *area,
                                 GtkCellRenderer *renderer,
                                 const gchar *attribute,
                                 gint column);
```

Connects an `attribute` to apply values from `column` for the GtkTreeModel in use.

Parameters

area	a GtkCellArea
renderer	the GtkCellRenderer to connect an attribute for
attribute	the attribute name
column	the GtkTreeModel column to fetch attribute values from

Since: 3.0

---

gtk_cell_area_attribute_disconnect ()

```
void
gtk_cell_area_attribute_disconnect (GtkCellArea *area,
                                    GtkCellRenderer *renderer,
                                    const gchar *attribute);
```

Disconnects `attribute` for the `renderer` in `area` so that attribute will no longer be updated with values from the model.

Parameters

area	a GtkCellArea
renderer	the GtkCellRenderer to disconnect an attribute for
attribute	the attribute name

Since: 3.0

---

gtk_cell_area_attribute_get_column ()

```
gint
gtk_cell_area_attribute_get_column (GtkCellArea *area,
                                    GtkCellRenderer *renderer,
                                    const gchar *attribute);
```

Returns the model column that an attribute has been mapped to, or -1 if the attribute is not mapped.

Parameters

area	a GtkCellArea
renderer	a GtkCellRenderer
attribute	an attribute on the renderer

Returns

the model column, or -1

Since: 3.14

---

gtk_cell_area_class_install_cell_property ()

```
void
gtk_cell_area_class_install_cell_property
                                (GtkCellAreaClass *aclass,
                                 guint property_id,
                                 GParamSpec *pspec);
```

Installs a cell property on a cell area class.

Parameters

aclass	a GtkCellAreaClass
property_id	the id for the property
pspec	the GParamSpec for the property

Since: 3.0

---

gtk_cell_area_class_find_cell_property ()

```
GParamSpec *
gtk_cell_area_class_find_cell_property
                                (GtkCellAreaClass *aclass,
                                 const gchar *property_name);
```

Finds a cell property of a cell area class by name.

Parameters

aclass	a GtkCellAreaClass
property_name	the name of the child property to find

Returns

the GParamSpec of the child property or `NULL` if `aclass` has no child property with that name.

[transfer none]

Since: 3.0

---

gtk_cell_area_class_list_cell_properties ()

```
GParamSpec **
gtk_cell_area_class_list_cell_properties
                                (GtkCellAreaClass *aclass,
                                 guint *n_properties);
```

Returns all cell properties of a cell area class.

Parameters

aclass	a GtkCellAreaClass
n_properties	location to return the number of cell properties found. [out]

Returns

a newly allocated `NULL`-terminated array of GParamSpec*. The array must be freed with `g_free()`.

[array length=n_properties][transfer container]

Since: 3.0

---

gtk_cell_area_add_with_properties ()

```
void
gtk_cell_area_add_with_properties (GtkCellArea *area,
                                   GtkCellRenderer *renderer,
                                   const gchar *first_prop_name,
                                   ...);
```

Adds `renderer` to `area` , setting cell properties at the same time. See `gtk_cell_area_add()` and `gtk_cell_area_cell_set()` for more details.

Parameters

area	a GtkCellArea

renderer	a GtkCellRenderer to be placed inside `area`
first_prop_name	the name of the first cell property to set
...	a `NULL`-terminated list of property names and values, starting with `first_prop_name`

Since: 3.0

---

gtk_cell_area_cell_set ()

```
void
gtk_cell_area_cell_set (GtkCellArea *area,
                        GtkCellRenderer *renderer,
                        const gchar *first_prop_name,
                        ...);
```

Sets one or more cell properties for `cell` in `area` .

Parameters

area	a GtkCellArea
renderer	a GtkCellRenderer which is a cell inside `area`
first_prop_name	the name of the first cell property to set
...	a `NULL`-terminated list of property names and values, starting with `first_prop_name`

Since: 3.0

---

gtk_cell_area_cell_get ()

```
void
gtk_cell_area_cell_get (GtkCellArea *area,
                        GtkCellRenderer *renderer,
                        const gchar *first_prop_name,
                        ...);
```

Gets the values of one or more cell properties for `renderer` in `area` .

Parameters

area	a GtkCellArea
renderer	a GtkCellRenderer which is inside `area`
first_prop_name	the name of the first cell property to get
...	return location for the first cell property, followed optionally by

	more name/return location pairs, followed by `NULL`

Since: 3.0

---

gtk_cell_area_cell_set_valist ()

```
void
gtk_cell_area_cell_set_valist (GtkCellArea *area,
                               GtkCellRenderer *renderer,
                               const gchar *first_property_name,
                               va_list var_args);
```

Sets one or more cell properties for `renderer` in `area` .

Parameters

area	a GtkCellArea
renderer	a GtkCellRenderer which inside `area`
first_property_name	the name of the first cell property to set
var_args	a `NULL`-terminated list of property names and values, starting with `first_prop_name`

Since: 3.0

---

gtk_cell_area_cell_get_valist ()

```
void
gtk_cell_area_cell_get_valist (GtkCellArea *area,
                               GtkCellRenderer *renderer,
                               const gchar *first_property_name,
                               va_list var_args);
```

Gets the values of one or more cell properties for `renderer` in `area` .

Parameters

area	a GtkCellArea
renderer	a GtkCellRenderer inside `area`
first_property_name	the name of the first property to get
var_args	return location for the first property, followed optionally by more name/return location pairs, followed by `NULL`

Since: 3.0

---

gtk_cell_area_cell_set_property ()

```
void
gtk_cell_area_cell_set_property (GtkCellArea *area,
                                 GtkCellRenderer *renderer,
                                 const gchar *property_name,
                                 const GValue *value);
```

Sets a cell property for `renderer` in `area` .

Parameters

area	a GtkCellArea
renderer	a GtkCellRenderer inside `area`
property_name	the name of the cell property to set
value	the value to set the cell property to

Since: 3.0

---

gtk_cell_area_cell_get_property ()

```
void
gtk_cell_area_cell_get_property (GtkCellArea *area,
                                 GtkCellRenderer *renderer,
                                 const gchar *property_name,
                                 GValue *value);
```

Gets the value of a cell property for `renderer` in `area` .

Parameters

area	a GtkCellArea
renderer	a GtkCellRenderer inside `area`
property_name	the name of the property to get
value	a location to return the value

Since: 3.0

---

gtk_cell_area_is_activatable ()

```
gboolean
gtk_cell_area_is_activatable (GtkCellArea *area);
```

Returns whether the area can do anything when activated, after applying new attributes to `area` .

Parameters

area	a GtkCellArea

Returns

whether `area` can do anything when activated.

Since: 3.0

---

gtk_cell_area_activate ()

```
gboolean
gtk_cell_area_activate (GtkCellArea *area,
                        GtkCellAreaContext *context,
                        GtkWidget *widget,
                        const GdkRectangle *cell_area,
                        GtkCellRendererState flags,
                        gboolean edit_only);
```

Activates `area` , usually by activating the currently focused cell, however some subclasses which embed widgets in the area can also activate a widget if it currently has the focus.

Parameters

area	a GtkCellArea
context	the GtkCellAreaContext in context with the current row data
widget	the GtkWidget that `area` is rendering on
cell_area	the size and location of `area` relative to `widget` 's allocation
flags	the GtkCellRendererState flags for `area` for this row of data.
edit_only	if `TRUE` then only cell renderers that are `GTK_CELL_RENDERER_MODE_EDITABLE` will be activated.

Returns

Whether `area` was successfully activated.

Since: 3.0

---

gtk_cell_area_focus ()

```
gboolean
gtk_cell_area_focus (GtkCellArea *area,
                     GtkDirectionType direction);
```

This should be called by the `area` 's owning layout widget when focus is to be passed to `area` , or moved within `area` for a given `direction` and row data.

Implementing GtkCellArea classes should implement this method to receive and navigate focus in its own way particular to how it lays out cells.

Parameters

area	a GtkCellArea
direction	the GtkDirectionType

Returns

`TRUE` if focus remains inside `area` as a result of this call.

Since: 3.0

---

gtk_cell_area_set_focus_cell ()

```
void
gtk_cell_area_set_focus_cell (GtkCellArea *area,
                              GtkCellRenderer *renderer);
```

Explicitly sets the currently focused cell to `renderer` .

This is generally called by implementations of `GtkCellAreaClass.focus()` or `GtkCellAreaClass.event()`, however it can also be used to implement functions such as `gtk_tree_view_set_cursor_on_cell()`.

Parameters

area	a GtkCellArea
renderer	the GtkCellRenderer to give focus to

Since: 3.0

---

gtk_cell_area_get_focus_cell ()

```
GtkCellRenderer *
gtk_cell_area_get_focus_cell (GtkCellArea *area);
```

Retrieves the currently focused cell for `area`

Parameters

area	a GtkCellArea

Returns

the currently focused cell in `area` .

[transfer none]

Since: 3.0

---

gtk_cell_area_add_focus_sibling ()

```
void
gtk_cell_area_add_focus_sibling (GtkCellArea *area,
                                 GtkCellRenderer *renderer,
                                 GtkCellRenderer *sibling);
```

Adds `sibling` to `renderer` 's focusable area, focus will be drawn around `renderer` and all of its siblings if `renderer` can focus for a given row.

Events handled by focus siblings can also activate the given focusable `renderer` .

Parameters

area	a GtkCellArea
renderer	the GtkCellRenderer expected to have focus
sibling	the GtkCellRenderer to add to `renderer` 's focus area

Since: 3.0

---

gtk_cell_area_remove_focus_sibling ()

```
void
gtk_cell_area_remove_focus_sibling (GtkCellArea *area,
                                    GtkCellRenderer *renderer,
                                    GtkCellRenderer *sibling);
```

Removes `sibling` from `renderer` 's focus sibling list (see `gtk_cell_area_add_focus_sibling()`).

Parameters

area	a GtkCellArea
renderer	the GtkCellRenderer expected to have focus
sibling	the GtkCellRenderer to remove from `renderer` 's focus area

Since: 3.0

---

gtk_cell_area_is_focus_sibling ()

```
gboolean
gtk_cell_area_is_focus_sibling (GtkCellArea *area,
                                GtkCellRenderer *renderer,
                                GtkCellRenderer *sibling);
```

Returns whether `sibling` is one of `renderer` 's focus siblings (see `gtk_cell_area_add_focus_sibling()`).

Parameters

area	a GtkCellArea
renderer	the GtkCellRenderer expected to have focus
sibling	the GtkCellRenderer to check against `renderer` 's sibling list

Returns

`TRUE` if `sibling` is a focus sibling of `renderer`

Since: 3.0

---

gtk_cell_area_get_focus_siblings ()

```
const GList *
gtk_cell_area_get_focus_siblings (GtkCellArea *area,
                                  GtkCellRenderer *renderer);
```

Gets the focus sibling cell renderers for `renderer` .

Parameters

area	a GtkCellArea
renderer	the GtkCellRenderer expected to have focus

Returns

A GList of GtkCellRenderers. The returned list is internal and should not be freed.

[element-type GtkCellRenderer][transfer none]

Since: 3.0

---

gtk_cell_area_get_focus_from_sibling ()

```
GtkCellRenderer *
gtk_cell_area_get_focus_from_sibling (GtkCellArea *area,
                                      GtkCellRenderer *renderer);
```

Gets the GtkCellRenderer which is expected to be focusable for which `renderer` is, or may be a sibling.

This is handy for GtkCellArea subclasses when handling events, after determining the renderer at the event location it can then chose to activate the focus cell for which the event cell may have been a sibling.

Parameters

area	a GtkCellArea

renderer	the GtkCellRenderer

Returns

the GtkCellRenderer for which `renderer` is a sibling, or `NULL`.

[nullable][transfer none]

Since: 3.0

---

gtk_cell_area_get_edited_cell ()

```
GtkCellRenderer *
gtk_cell_area_get_edited_cell (GtkCellArea *area);
```

Gets the GtkCellRenderer in `area` that is currently being edited.

Parameters

area	a GtkCellArea

Returns

The currently edited GtkCellRenderer.

[transfer none]

Since: 3.0

---

gtk_cell_area_get_edit_widget ()

```
GtkCellEditable *
gtk_cell_area_get_edit_widget (GtkCellArea *area);
```

Gets the GtkCellEditable widget currently used to edit the currently edited cell.

Parameters

area	a GtkCellArea

Returns

The currently active GtkCellEditable widget.

[transfer none]

Since: 3.0

---

gtk_cell_area_activate_cell ()

```
gboolean
gtk_cell_area_activate_cell (GtkCellArea *area,
                             GtkWidget *widget,
                             GtkCellRenderer *renderer,
                             GdkEvent *event,
                             const GdkRectangle *cell_area,
                             GtkCellRendererState flags);
```

This is used by GtkCellArea subclasses when handling events to activate cells, the base GtkCellArea class activates cells for keyboard events for free in its own GtkCellArea->`activate()` implementation.

Parameters

area	a GtkCellArea
widget	the GtkWidget that `area` is rendering onto
renderer	the GtkCellRenderer in `area` to activate
event	the GdkEvent for which cell activation should occur
cell_area	the GdkRectangle in `widget` relative coordinates of `renderer` for the current row.
flags	the GtkCellRendererState for `renderer`

Returns

whether cell activation was successful

Since: 3.0

---

gtk_cell_area_stop_editing ()

```
void
gtk_cell_area_stop_editing (GtkCellArea *area,
                            gboolean canceled);
```

Explicitly stops the editing of the currently edited cell.

If `canceled` is `TRUE`, the currently edited cell renderer will emit the ::editing-canceled signal, otherwise the the ::editing-done signal will be emitted on the current edit widget.

See `gtk_cell_area_get_edited_cell()` and `gtk_cell_area_get_edit_widget()`.

Parameters

area	a GtkCellArea
canceled	whether editing was canceled.

Since: 3.0

---

gtk_cell_area_inner_cell_area ()

```
void
gtk_cell_area_inner_cell_area (GtkCellArea *area,
                               GtkWidget *widget,
                               const GdkRectangle *cell_area,
                               GdkRectangle *inner_area);
```

This is a convenience function for GtkCellArea implementations to get the inner area where a given GtkCellRenderer will be rendered. It removes any padding previously added by `gtk_cell_area_request_renderer()`.

Parameters

area	a GtkCellArea	
widget	the GtkWidget that `area` is rendering onto	
cell_area	the `widget` relative coordinates where one of `area` 's cells is to be placed	
inner_area	the return location for the inner cell area.	[out]

Since: 3.0

---

gtk_cell_area_request_renderer ()

```
void
gtk_cell_area_request_renderer (GtkCellArea *area,
                                GtkCellRenderer *renderer,
                                GtkOrientation orientation,
                                GtkWidget *widget,
                                gint for_size,
                                gint *minimum_size,
                                gint *natural_size);
```

This is a convenience function for GtkCellArea implementations to request size for cell renderers. It's important to use this function to request size and then use `gtk_cell_area_inner_cell_area()` at render and event time since this function will add padding around the cell for focus painting.

Parameters

area	a GtkCellArea
renderer	the GtkCellRenderer to request size for
orientation	the GtkOrientation in which to request size
widget	the GtkWidget that `area` is rendering onto
for_size	the allocation contextual size to request for, or -1 if the base request

	for the orientation is to be returned.	
minimum_size	location to store the minimum size, or NULL.	[out][allow-none]
natural_size	location to store the natural size, or NULL.	[out][allow-none]

Since: 3.0

Types and Values

struct GtkCellArea

```
struct GtkCellArea;
```

---

struct GtkCellAreaClass

```
struct GtkCellAreaClass {
  /* Basic methods */
  void               (* add)                          (GtkCellArea             *area,
                                                       GtkCellRenderer
*renderer);
  void               (* remove)                       (GtkCellArea             *area,
                                                       GtkCellRenderer
*renderer);
  void               (* foreach)                      (GtkCellArea             *area,
                                                       GtkCellCallback
callback,
                                                       gpointer
callback_data);
  void               (* foreach_alloc)                (GtkCellArea             *area,
                                                       GtkCellAreaContext
*context,
                                                       GtkWidget               *widget,
                                                       const GdkRectangle
*cell_area,
                                                       const GdkRectangle
*background_area,
                                                       GtkCellAllocCallback
callback,
                                                       gpointer
callback_data);
  gint               (* event)                        (GtkCellArea             *area,
                                                       GtkCellAreaContext
*context,
                                                       GtkWidget               *widget,
                                                       GdkEvent                *event,
                                                       const GdkRectangle
*cell_area,
                                                       GtkCellRendererState     flags);
  void               (* render)                       (GtkCellArea             *area,
                                                       GtkCellAreaContext
*context,
                                                       GtkWidget               *widget,
                                                       cairo_t                 *cr,
                                                       const GdkRectangle
```

```
*background_area,
                                                        const GdkRectangle
*cell_area,
                                                        GtkCellRendererState     flags,
                                                        gboolean
paint_focus);
  void               (* apply_attributes)               (GtkCellArea             *area,
                                                        GtkTreeModel
*tree_model,
                                                        GtkTreeIter             *iter,
                                                        gboolean
is_expander,
                                                        gboolean
is_expanded);

  /* Geometry */
  GtkCellAreaContext *(* create_context)                (GtkCellArea             *area);
  GtkCellAreaContext *(* copy_context)                  (GtkCellArea             *area,
                                                        GtkCellAreaContext
*context);
  GtkSizeRequestMode (* get_request_mode)               (GtkCellArea             *area);
  void               (* get_preferred_width)            (GtkCellArea             *area,
                                                        GtkCellAreaContext
*context,
                                                        GtkWidget               *widget,
                                                        gint
*minimum_width,
                                                        gint
*natural_width);
  void               (* get_preferred_height_for_width) (GtkCellArea             *area,
                                                        GtkCellAreaContext
*context,
                                                        GtkWidget               *widget,
                                                        gint                     width,
                                                        gint
*minimum_height,
                                                        gint
*natural_height);
  void               (* get_preferred_height)           (GtkCellArea             *area,
                                                        GtkCellAreaContext
*context,
                                                        GtkWidget               *widget,
                                                        gint
*minimum_height,
                                                        gint
*natural_height);
  void               (* get_preferred_width_for_height) (GtkCellArea             *area,
                                                        GtkCellAreaContext
*context,
                                                        GtkWidget               *widget,
                                                        gint                     height,
                                                        gint
*minimum_width,
                                                        gint
*natural_width);

  /* Cell Properties */
  void               (* set_cell_property)              (GtkCellArea             *area,
                                                        GtkCellRenderer
*renderer,
                                                        guint
property_id,
                                                        const GValue            *value,
```

```
                                                      GParamSpec              *pspec);
  void                (* get_cell_property)          (GtkCellArea             *area,
                                                      GtkCellRenderer
*renderer,
                                                      guint
property_id,
                                                      GValue                  *value,
                                                      GParamSpec              *pspec);

  /* Focus */
  gboolean            (* focus)                      (GtkCellArea             *area,
                                                      GtkDirectionType
direction);
  gboolean            (* is_activatable)             (GtkCellArea             *area);
  gboolean            (* activate)                   (GtkCellArea             *area,
                                                      GtkCellAreaContext
*context,
                                                      GtkWidget               *widget,
                                                      const GdkRectangle
*cell_area,
                                                      GtkCellRendererState     flags,
                                                      gboolean
edit_only);
};
```

Members

`add` ()	adds a GtkCellRenderer to the area.
`remove` ()	removes a GtkCellRenderer from the area.
`foreach` ()	calls the GtkCellCallback function on every GtkCellRenderer in the area with the provided user data until the callback returns `TRUE`.
`foreach_alloc` ()	Calls the GtkCellAllocCallback function on every GtkCellRenderer in the area with the allocated area for the cell and the provided user data until the callback returns `TRUE`.
`event` ()	Handle an event in the area, this is generally used to activate a cell at the event location for button events but can also be used to generically pass events to GtkWidgets drawn onto the area.
`render` ()	Actually render the area’s cells to the specified rectangle, `background_area` should be correctly distributed to the cells corresponding background areas.
`apply_attributes` ()	Apply the cell attributes to the cells. This is implemented as a signal and generally GtkCellArea subclasses don't need to implement it since it is handled by the base class.

`create_context` ()	Creates and returns a class specific GtkCellAreaContext to store cell alignment and allocation details for a said GtkCellArea class.
`copy_context` ()	Creates a new GtkCellAreaContext in the same state as the passed `context` with any cell alignment data and allocations intact.
`get_request_mode` ()	This allows an area to tell its layouting widget whether it prefers to be allocated in `GTK_SIZE_REQUEST_HEIGHT_FOR_WIDTH` or `GTK_SIZE_REQUEST_WIDTH_FOR_HEIGHT` mode.
`get_preferred_width` ()	Calculates the minimum and natural width of the areas cells with the current attributes applied while considering the particular layouting details of the said GtkCellArea. While requests are performed over a series of rows, alignments and overall minimum and natural sizes should be stored in the corresponding GtkCellAreaContext.
`get_preferred_height_for_width` ()	Calculates the minimum and natural height for the area if the passed `context` would be allocated the given width. When implementing this virtual method it is safe to assume that `context` has already stored the aligned cell widths for every GtkTreeModel row that `context` will be allocated for since this information was stored at `GtkCellAreaClass.get_preferred_width()` time. This virtual method should also store any necessary alignments of cell heights for the case that the context is allocated a height.
`get_preferred_height` ()	Calculates the minimum and natural height of the areas cells with the current attributes applied. Essentially this is the same as `GtkCellAreaClass.get_preferred_width()` only for areas that are being requested as `GTK_SIZE_REQUEST_WIDTH_FOR_HEIGHT`.
`get_preferred_width_for_height` ()	Calculates the minimum and natural width for the area if the passed

	`context` would be allocated the given height. The same as `GtkCellAreaClass.get_preferred_height_for_width()` only for handling requests in the `GTK_SIZE_REQUEST_WIDTH_FOR_HEIGHT` mode.
`set_cell_property` ()	This should be implemented to handle changes in child cell properties for a given GtkCellRenderer that were previously installed on the GtkCellAreaClass with `gtk_cell_area_class_install_cell_property()`.
`get_cell_property` ()	This should be implemented to report the values of child cell properties for a given child GtkCellRenderer.
`focus` ()	This virtual method should be implemented to navigate focus from cell to cell inside the GtkCellArea. The GtkCellArea should move focus from cell to cell inside the area and return `FALSE` if focus logically leaves the area with the following exceptions: When the area contains no activatable cells, the entire area recieves focus. Focus should not be given to cells that are actually “focus siblings” of other sibling cells (see `gtk_cell_area_get_focus_from_sibling()`). Focus is set by calling `gtk_cell_area_set_focus_cell()`.
`is_activatable` ()	Returns whether the GtkCellArea can respond to `GtkCellAreaClass.activate()`, usually this does not need to be implemented since the base class takes care of this however it can be enhanced if the GtkCellArea subclass can handle activation in other ways than activating its GtkCellRenderers.
`activate` ()	This is called when the layouting widget rendering the GtkCellArea activates the focus cell (see `gtk_cell_area_get_focus_cell()`).

Property Details

The “edit-widget” property

```
  “edit-widget”              GtkCellEditable *
```

The widget currently editing the edited cell

This property is read-only and only changes as a result of a call `gtk_cell_area_activate_cell()`.

Flags: Read

Since: 3.0

The “edited-cell” property

```
  “edited-cell”              GtkCellRenderer *
```

The cell in the area that is currently edited

This property is read-only and only changes as a result of a call `gtk_cell_area_activate_cell()`.

Flags: Read

Since: 3.0

The “focus-cell” property

```
  “focus-cell”               GtkCellRenderer *
```

The cell in the area that currently has focus

Flags: Read / Write

Since: 3.0

Signal Details

The “add-editable” signal

```
void
user_function (GtkCellArea     *area,
               GtkCellRenderer *renderer,
               GtkCellEditable *editable,
               GdkRectangle    *cell_area,
               gchar           *path,
               gpointer         user_data)
```

Indicates that editing has started on `renderer` and that `editable` should be added to the owning cell-layouting widget at `cell_area` .

Parameters

area	the GtkCellArea where editing started
renderer	the GtkCellRenderer that started the edited
editable	the GtkCellEditable widget to add
cell_area	the GtkWidget relative GdkRectangle coordinates where `editable` should be added
path	the GtkTreePath string this edit was initiated for
user_data	user data set when the signal handler was connected.

Flags: Run First

Since: 3.0

The `"apply-attributes"` signal

```
void
user_function (GtkCellArea  *area,
               GtkTreeModel *model,
               GtkTreeIter  *iter,
               gboolean      is_expander,
               gboolean      is_expanded,
               gpointer      user_data)
```

This signal is emitted whenever applying attributes to `area` from `model`

Parameters

area	the GtkCellArea to apply the attributes to
model	the GtkTreeModel to apply the attributes from
iter	the GtkTreeIter indicating which row to apply the attributes of
is_expander	whether the view shows children for this row
is_expanded	whether the view is currently showing the children of this row
user_data	user data set when the signal handler was connected.

Flags: Run First

Since: 3.0

The “focus-changed” signal

```
void
user_function (GtkCellArea     *area,
               GtkCellRenderer *renderer,
               gchar           *path,
               gpointer         user_data)
```

Indicates that focus changed on this `area` . This signal is emitted either as a result of focus handling or event handling.

It's possible that the signal is emitted even if the currently focused renderer did not change, this is because focus may change to the same renderer in the same cell area for a different row of data.

Parameters

area	the GtkCellArea where focus changed
renderer	the GtkCellRenderer that has focus
path	the current GtkTreePath string set for `area`
user_data	user data set when the signal handler was connected.

Flags: Run First

Since: 3.0

The “remove-editable” signal

```
void
user_function (GtkCellArea     *area,
               GtkCellRenderer *renderer,
               GtkCellEditable *editable,
               gpointer         user_data)
```

Indicates that editing finished on `renderer` and that `editable` should be removed from the owning cell-layouting widget.

Parameters

area	the GtkCellArea where editing finished
renderer	the GtkCellRenderer that finished editeding
editable	the GtkCellEditable widget to remove
user_data	user data set when the signal handler was connected.

Flags: Run First

Since: 3.0

GtkCellAreaBox

GtkCellAreaBox — A cell area that renders GtkCellRenderers into a row or a column

Functions

GtkCellArea *	gtk_cell_area_box_new ()
void	gtk_cell_area_box_pack_start ()
void	gtk_cell_area_box_pack_end ()
gint	gtk_cell_area_box_get_spacing ()
void	gtk_cell_area_box_set_spacing ()

Properties

gint	spacing	Read / Write

Child Properties

gboolean	align	Read / Write
gboolean	expand	Read / Write
gboolean	fixed-size	Read / Write
GtkPackType	pack-type	Read / Write

Types and Values

struct	GtkCellAreaBox
struct	GtkCellAreaBoxClass

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkCellArea
        └── GtkCellAreaBox
```

Implemented Interfaces

GtkCellAreaBox implements GtkCellLayout, GtkBuildable and GtkOrientable.

Includes

```
#include <gtk/gtk.h>
```

Description

The GtkCellAreaBox renders cell renderers into a row or a column depending on its GtkOrientation.

GtkCellAreaBox uses a notion of packing. Packing refers to adding cell renderers with reference to a particular position in a GtkCellAreaBox. There are two reference positions: the start and the end of the box. When the GtkCellAreaBox is oriented in the `GTK_ORIENTATION_VERTICAL` orientation, the start is defined as the top of the box and the end is defined as the bottom. In the `GTK_ORIENTATION_HORIZONTAL` orientation start is defined as the left side and the end is defined as the right side.

Alignments of GtkCellRenderers rendered in adjacent rows can be configured by configuring the GtkCellAreaBox align child cell property with `gtk_cell_area_cell_set_property()` or by specifying the "align" argument to `gtk_cell_area_box_pack_start()` and `gtk_cell_area_box_pack_end()`.

Functions

gtk_cell_area_box_new ()

```
GtkCellArea *
gtk_cell_area_box_new (void);
```

Creates a new GtkCellAreaBox.

Returns

a newly created GtkCellAreaBox

Since: 3.0

---

gtk_cell_area_box_pack_start ()

```
void
gtk_cell_area_box_pack_start (GtkCellAreaBox *box,
                              GtkCellRenderer *renderer,
                              gboolean expand,
                              gboolean align,
                              gboolean fixed);
```

Adds `renderer` to `box` , packed with reference to the start of `box` .

The `renderer` is packed after any other GtkCellRenderer packed with reference to the start of `box` .

Parameters

box	a GtkCellAreaBox
renderer	the GtkCellRenderer to add
expand	whether `renderer` should receive extra space when the area receives more than its natural size
align	whether `renderer` should be aligned in adjacent rows
fixed	whether `renderer` should have the same size in all rows

Since: 3.0

---

gtk_cell_area_box_pack_end ()

```
void
gtk_cell_area_box_pack_end (GtkCellAreaBox *box,
                            GtkCellRenderer *renderer,
                            gboolean expand,
                            gboolean align,
                            gboolean fixed);
```

Adds `renderer` to `box` , packed with reference to the end of `box` .

The `renderer` is packed after (away from end of) any other GtkCellRenderer packed with reference to the end of `box` .

Parameters

box	a GtkCellAreaBox
renderer	the GtkCellRenderer to add
expand	whether `renderer` should receive extra space when the area receives more than its natural size
align	whether `renderer` should be aligned in adjacent rows
fixed	whether `renderer` should have the same size in all rows

Since: 3.0

---

gtk_cell_area_box_get_spacing ()

```
gint
gtk_cell_area_box_get_spacing (GtkCellAreaBox *box);
```

Gets the spacing added between cell renderers.

Parameters

box	a GtkCellAreaBox

Returns

the space added between cell renderers in `box` .

Since: 3.0

---

gtk_cell_area_box_set_spacing ()

```
void
gtk_cell_area_box_set_spacing (GtkCellAreaBox *box,
                               gint spacing);
```

Sets the spacing to add between cell renderers in `box` .

Parameters

box	a GtkCellAreaBox
spacing	the space to add between GtkCellRenderers

Since: 3.0

Types and Values

struct GtkCellAreaBox

```
struct GtkCellAreaBox;
```

---

struct GtkCellAreaBoxClass

```
struct GtkCellAreaBoxClass {
};
```

Property Details

The “spacing” property

```
  “spacing”                  gint
```

The amount of space to reserve between cells.

Flags: Read / Write

Allowed values: >= 0

Default value: 0

Since: 3.0

Child Property Details

The “align” child property

```
  “align”                    gboolean
```

Whether the cell renderer should be aligned in adjacent rows.

Flags: Read / Write

Default value: FALSE

Since: 3.0

The “expand” child property

“expand” gboolean

Whether the cell renderer should receive extra space when the area receives more than its natural size.

Flags: Read / Write

Default value: FALSE

Since: 3.0

The “fixed-size” child property

“fixed-size” gboolean

Whether the cell renderer should require the same size for all rows for which it was requested.

Flags: Read / Write

Default value: TRUE

Since: 3.0

The “pack-type” child property

“pack-type” GtkPackType

A GtkPackType indicating whether the cell renderer is packed with reference to the start or end of the area.

Flags: Read / Write

Default value: GTK_PACK_START

Since: 3.0

GtkCellAreaContext

GtkCellAreaContext — Stores geometrical information for a series of rows in a GtkCellArea

Functions

GtkCellArea *	gtk_cell_area_context_get_area ()
void	gtk_cell_area_context_allocate ()
void	gtk_cell_area_context_reset ()
void	gtk_cell_area_context_get_preferred_width ()
void	gtk_cell_area_context_get_preferred_height ()
void	gtk_cell_area_context_get_preferred_height_for_width ()
void	gtk_cell_area_context_get_preferred_width_for_height ()
void	gtk_cell_area_context_get_allocation ()
void	gtk_cell_area_context_push_preferred_width ()
void	gtk_cell_area_context_push_preferred_height ()

Properties

GtkCellArea *	area	Read / Write / Construct Only
gint	minimum-height	Read
gint	minimum-width	Read
gint	natural-height	Read
gint	natural-width	Read

Types and Values

struct	GtkCellAreaContextClass
	GtkCellAreaContext

Object Hierarchy

```
GObject
└── GtkCellAreaContext
```

Includes

```
#include <gtk/gtk.h>
```

Description

The GtkCellAreaContext object is created by a given GtkCellArea implementation via its `GtkCellAreaClass.create_context()` virtual method and is used to store cell sizes and alignments for a series of GtkTreeModel rows that are requested and rendered in the same context.

GtkCellLayout widgets can create any number of contexts in which to request and render groups of data rows. However, it's important that the same context which was used to request sizes for a given GtkTreeModel row also be used for the same row when calling other GtkCellArea APIs such as `gtk_cell_area_render()` and `gtk_cell_area_event()`.

Functions

gtk_cell_area_context_get_area ()

```
GtkCellArea *
gtk_cell_area_context_get_area (GtkCellAreaContext *context);
```

Fetches the GtkCellArea this `context` was created by.

This is generally unneeded by layouting widgets; however, it is important for the context implementation itself to fetch information about the area it is being used for.

For instance at `GtkCellAreaContextClass.allocate()` time it's important to know details about any cell spacing that the GtkCellArea is configured with in order to compute a proper allocation.

Parameters

context	a GtkCellAreaContext

Returns

the GtkCellArea this context was created by.

[transfer none]

Since: 3.0

---

gtk_cell_area_context_allocate ()

```
void
gtk_cell_area_context_allocate (GtkCellAreaContext *context,
                                gint width,
                                gint height);
```

Allocates a width and/or a height for all rows which are to be rendered with `context` .

Usually allocation is performed only horizontally or sometimes vertically since a group of rows are usually rendered side by side vertically or horizontally and share either the same width or the same height. Sometimes they are allocated in both horizontal and vertical orientations producing a homogeneous effect of the rows. This is generally the case for GtkTreeView when "fixed-height-mode" is enabled.

Since 3.0

Parameters

context	a GtkCellAreaContext
width	the allocated width for all GtkTreeModel rows rendered with `context` , or -1.
height	the allocated height for all GtkTreeModel rows rendered with `context` , or -1.

gtk_cell_area_context_reset ()

```
void
gtk_cell_area_context_reset (GtkCellAreaContext *context);
```

Resets any previously cached request and allocation data.

When underlying GtkTreeModel data changes its important to reset the context if the content size is allowed to shrink. If the content size is only allowed to grow (this is usually an option for views rendering large data stores as a measure of optimization), then only the row that changed or was inserted needs to be (re)requested with `gtk_cell_area_get_preferred_width()`.

When the new overall size of the context requires that the allocated size changes (or whenever this allocation changes at all), the variable row sizes need to be re-requested for every row.

For instance, if the rows are displayed all with the same width from top to bottom then a change in the allocated width necessitates a recalculation of all the displayed row heights using `gtk_cell_area_get_preferred_height_for_width()`.

Since 3.0

Parameters

context	a GtkCellAreaContext	

---

gtk_cell_area_context_get_preferred_width ()

```
void
gtk_cell_area_context_get_preferred_width
                                (GtkCellAreaContext *context,
                                 gint *minimum_width,
                                 gint *natural_width);
```

Gets the accumulative preferred width for all rows which have been requested with this context.

After `gtk_cell_area_context_reset()` is called and/or before ever requesting the size of a GtkCellArea, the returned values are 0.

Parameters

context	a GtkCellAreaContext	
minimum_width	location to store the minimum width, or `NULL`.	[out][allow-none]
natural_width	location to store the natural width, or `NULL`.	[out][allow-none]

Since: 3.0

---

gtk_cell_area_context_get_preferred_height ()

```
void
gtk_cell_area_context_get_preferred_height
                                (GtkCellAreaContext *context,
                                 gint *minimum_height,
                                 gint *natural_height);
```

Gets the accumulative preferred height for all rows which have been requested with this context.

After `gtk_cell_area_context_reset()` is called and/or before ever requesting the size of a GtkCellArea, the returned values are 0.

Parameters

context	a GtkCellAreaContext	
minimum_height	location to store the minimum height, or `NULL`.	[out][allow-none]
natural_height	location to store the natural height, or `NULL`.	[out][allow-none]

Since: 3.0

---

gtk_cell_area_context_get_preferred_height_for_width ()

```
void
gtk_cell_area_context_get_preferred_height_for_width
                                (GtkCellAreaContext *context,
                                 gint width,
                                 gint *minimum_height,
                                 gint *natural_height);
```

Gets the accumulative preferred height for `width` for all rows which have been requested for the same said `width` with this context.

After `gtk_cell_area_context_reset()` is called and/or before ever requesting the size of a GtkCellArea, the returned values are -1.

Parameters

context	a GtkCellAreaContext	
width	a proposed width for allocation	
minimum_height	location to store the minimum height, or `NULL`.	[out][allow-none]
natural_height	location to store the natural height, or `NULL`.	[out][allow-none]

Since: 3.0

---

gtk_cell_area_context_get_preferred_width_for_height ()

```
void
gtk_cell_area_context_get_preferred_width_for_height
```

```
                                (GtkCellAreaContext *context,
                                 gint height,
                                 gint *minimum_width,
                                 gint *natural_width);
```

Gets the accumulative preferred width for `height` for all rows which have been requested for the same said `height` with this context.

After `gtk_cell_area_context_reset()` is called and/or before ever requesting the size of a GtkCellArea, the returned values are -1.

Parameters

context	a GtkCellAreaContext	
height	a proposed height for allocation	
minimum_width	location to store the minimum width, or `NULL`.	[out][allow-none]
natural_width	location to store the natural width, or `NULL`.	[out][allow-none]

Since: 3.0

---

gtk_cell_area_context_get_allocation ()

```
void
gtk_cell_area_context_get_allocation (GtkCellAreaContext *context,
                                      gint *width,
                                      gint *height);
```

Fetches the current allocation size for `context` .

If the context was not allocated in width or height, or if the context was recently reset with `gtk_cell_area_context_reset()`, the returned value will be -1.

Parameters

context	a GtkCellAreaContext	
width	location to store the allocated width, or `NULL`.	[out][allow-none]
height	location to store the allocated height, or `NULL`.	[out][allow-none]

Since: 3.0

---

gtk_cell_area_context_push_preferred_width ()

```
void
gtk_cell_area_context_push_preferred_width
                                (GtkCellAreaContext *context,
                                 gint minimum_width,
                                 gint natural_width);
```

Causes the minimum and/or natural width to grow if the new proposed sizes exceed the current minimum and

natural width.

This is used by GtkCellAreaContext implementations during the request process over a series of GtkTreeModel rows to progressively push the requested width over a series of `gtk_cell_area_get_preferred_width()` requests.

Parameters

context	a GtkCellAreaContext
minimum_width	the proposed new minimum width for `context`
natural_width	the proposed new natural width for `context`

Since: 3.0

---

gtk_cell_area_context_push_preferred_height ()

```
void
gtk_cell_area_context_push_preferred_height
                                (GtkCellAreaContext *context,
                                 gint minimum_height,
                                 gint natural_height);
```

Causes the minimum and/or natural height to grow if the new proposed sizes exceed the current minimum and natural height.

This is used by GtkCellAreaContext implementations during the request process over a series of GtkTreeModel rows to progressively push the requested height over a series of `gtk_cell_area_get_preferred_height()` requests.

Parameters

context	a GtkCellAreaContext
minimum_height	the proposed new minimum height for `context`
natural_height	the proposed new natural height for `context`

Since: 3.0

Types and Values

struct GtkCellAreaContextClass

```
struct GtkCellAreaContextClass {
  void    (* allocate)                      (GtkCellAreaContext *context,
                                             gint                width,
                                             gint                height);
  void    (* reset)                         (GtkCellAreaContext *context);
  void    (* get_preferred_height_for_width) (GtkCellAreaContext *context,
```

```
                                                gint                 width,
                                                gint                *minimum_height,
                                                gint                *natural_height);
  void    (* get_preferred_width_for_height) (GtkCellAreaContext *context,
                                                gint                 height,
                                                gint                *minimum_width,
                                                gint                *natural_width);
};
```

Members

`allocate` ()	This tells the context that an allocation width or height (or both) have been decided for a group of rows. The context should store any allocations for internally aligned cells at this point so that they dont need to be recalculated at `gtk_cell_area_render()` time.
`reset` ()	Clear any previously stored information about requested and allocated sizes for the context.
`get_preferred_height_for_width` ()	Returns the aligned height for the given width that context must store while collecting sizes for it's rows.
`get_preferred_width_for_height` ()	Returns the aligned width for the given height that context must store while collecting sizes for it's rows.

GtkCellAreaContext

```
typedef struct _GtkCellAreaContext GtkCellAreaContext;
```

Property Details

The `"area"` property

```
  "area"                     GtkCellArea *
```

The GtkCellArea this context was created by

Flags: Read / Write / Construct Only

Since: 3.0

The `"minimum-height"` property

```
  "minimum-height"           gint
```

The minimum height for the GtkCellArea in this context for all GtkTreeModel rows that this context was

requested for using `gtk_cell_area_get_preferred_height()`.

Flags: Read

Allowed values: >= -1

Default value: -1

Since: 3.0

The “minimum-width” property

“minimum-width” gint

The minimum width for the GtkCellArea in this context for all GtkTreeModel rows that this context was requested for using `gtk_cell_area_get_preferred_width()`.

Flags: Read

Allowed values: >= -1

Default value: -1

Since: 3.0

The “natural-height” property

“natural-height” gint

The natural height for the GtkCellArea in this context for all GtkTreeModel rows that this context was requested for using `gtk_cell_area_get_preferred_height()`.

Flags: Read

Allowed values: >= -1

Default value: -1

Since: 3.0

The “natural-width” property

“natural-width” gint

The natural width for the GtkCellArea in this context for all GtkTreeModel rows that this context was requested for using `gtk_cell_area_get_preferred_width()`.

Flags: Read

Allowed values: >= -1

Default value: -1

Since: 3.0

GtkCellRenderer

GtkCellRenderer — An object for rendering a single cell

Functions

void	gtk_cell_renderer_class_set_accessible_type ()
void	gtk_cell_renderer_get_aligned_area ()
void	gtk_cell_renderer_get_size ()
void	gtk_cell_renderer_render ()
gboolean	gtk_cell_renderer_activate ()
GtkCellEditable *	gtk_cell_renderer_start_editing ()
void	gtk_cell_renderer_stop_editing ()
void	gtk_cell_renderer_get_fixed_size ()
void	gtk_cell_renderer_set_fixed_size ()
gboolean	gtk_cell_renderer_get_visible ()
void	gtk_cell_renderer_set_visible ()
gboolean	gtk_cell_renderer_get_sensitive ()
void	gtk_cell_renderer_set_sensitive ()
void	gtk_cell_renderer_get_alignment ()
void	gtk_cell_renderer_set_alignment ()
void	gtk_cell_renderer_get_padding ()
void	gtk_cell_renderer_set_padding ()
GtkStateFlags	gtk_cell_renderer_get_state ()
gboolean	gtk_cell_renderer_is_activatable ()
void	gtk_cell_renderer_get_preferred_height ()
void	gtk_cell_renderer_get_preferred_height_for_width ()
void	gtk_cell_renderer_get_preferred_size ()
void	gtk_cell_renderer_get_preferred_width ()
void	gtk_cell_renderer_get_preferred_width_for_height ()
GtkSizeRequestMode	gtk_cell_renderer_get_request_mode ()

Properties

gchar *	cell-background	Write
GdkColor *	cell-background-gdk	Read / Write
GdkRGBA *	cell-background-rgba	Read / Write
gboolean	cell-background-set	Read / Write
gboolean	editing	Read
gint	height	Read / Write
gboolean	is-expanded	Read / Write
gboolean	is-expander	Read / Write
GtkCellRendererMode	mode	Read / Write
gboolean	sensitive	Read / Write
gboolean	visible	Read / Write
gint	width	Read / Write
gfloat	xalign	Read / Write
guint	xpad	Read / Write

gfloat	yalign	Read / Write
guint	ypad	Read / Write

Signals

void	editing-canceled	Run First
void	editing-started	Run First

Types and Values

enum	GtkCellRendererState
enum	GtkCellRendererMode
struct	GtkCellRenderer
struct	GtkCellRendererClass

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkCellRenderer
        ├── GtkCellRendererText
        ├── GtkCellRendererPixbuf
        ├── GtkCellRendererProgress
        ├── GtkCellRendererSpinner
        └── GtkCellRendererToggle
```

Includes

```
#include <gtk/gtk.h>
```

Description

The GtkCellRenderer is a base class of a set of objects used for rendering a cell to a cairo_t. These objects are used primarily by the GtkTreeView widget, though they aren't tied to them in any specific way. It is worth noting that GtkCellRenderer is not a GtkWidget and cannot be treated as such.

The primary use of a GtkCellRenderer is for drawing a certain graphical elements on a cairo_t. Typically, one cell renderer is used to draw many cells on the screen. To this extent, it isn't expected that a CellRenderer keep any permanent state around. Instead, any state is set just prior to use using GObjects property system. Then, the cell is measured using `gtk_cell_renderer_get_size()`. Finally, the cell is rendered in the correct location using `gtk_cell_renderer_render()`.

There are a number of rules that must be followed when writing a new GtkCellRenderer. First and foremost, it's important that a certain set of properties will always yield a cell renderer of the same size, barring a GtkStyle change. The GtkCellRenderer also has a number of generic properties that are expected to be honored by all children.

Beyond merely rendering a cell, cell renderers can optionally provide active user interface elements. A cell renderer can be "activatable" like GtkCellRendererToggle, which toggles when it gets activated by a mouse click, or it can be "editable" like GtkCellRendererText, which allows the user to edit the text using a widget implementing the GtkCellEditable interface, e.g. GtkEntry. To make a cell renderer activatable or editable, you have to implement the GtkCellRendererClass.activate or GtkCellRendererClass.start_editing virtual functions,

respectively.

Many properties of GtkCellRenderer and its subclasses have a corresponding “set” property, e.g. “cell-background-set” corresponds to “cell-background”. These “set” properties reflect whether a property has been set or not. You should not set them independently.

Functions

gtk_cell_renderer_class_set_accessible_type ()

```
void
gtk_cell_renderer_class_set_accessible_type
                                (GtkCellRendererClass *renderer_class,
                                 GType type);
```

Sets the type to be used for creating accessibles for cells rendered by cell renderers of `renderer_class` . Note that multiple accessibles will be created.

This function should only be called from class init functions of cell renderers.

Parameters

renderer_class	class to set the accessible type for
type	The object type that implements the accessible for `widget_class` . The type must be a subtype of GtkRendererCellAccessible

---

gtk_cell_renderer_get_aligned_area ()

```
void
gtk_cell_renderer_get_aligned_area (GtkCellRenderer *cell,
                                    GtkWidget *widget,
                                    GtkCellRendererState flags,
                                    const GdkRectangle *cell_area,
                                    GdkRectangle *aligned_area);
```

Gets the aligned area used by `cell` inside `cell_area` . Used for finding the appropriate edit and focus rectangle.

Parameters

cell	a GtkCellRenderer instance	
widget	the GtkWidget this cell will be rendering to	
flags	render flags	
cell_area	cell area which would be passed to `gtk_cell_renderer_render()`	
aligned_area	the return location for the space	[out]

	inside `cell_area` that would acually be used to render.	

Since: 3.0

---

gtk_cell_renderer_get_size ()

```
void
gtk_cell_renderer_get_size (GtkCellRenderer *cell,
                            GtkWidget *widget,
                            const GdkRectangle *cell_area,
                            gint *x_offset,
                            gint *y_offset,
                            gint *width,
                            gint *height);
```

`gtk_cell_renderer_get_size` has been deprecated since version 3.0 and should not be used in newly-written code.

Use `gtk_cell_renderer_get_preferred_size()` instead.

Obtains the width and height needed to render the cell. Used by view widgets to determine the appropriate size for the cell_area passed to `gtk_cell_renderer_render()`. If `cell_area` is not `NULL`, fills in the x and y offsets (if set) of the cell relative to this location.

Please note that the values set in `width` and `height` , as well as those in `x_offset` and `y_offset` are inclusive of the xpad and ypad properties.

Parameters

cell	a GtkCellRenderer	
widget	the widget the renderer is rendering to	
cell_area	The area a cell will be allocated, or `NULL`.	[allow-none]
x_offset	location to return x offset of cell relative to `cell_area` , or `NULL`.	[out][allow-none]
y_offset	location to return y offset of cell relative to `cell_area` , or `NULL`.	[out][allow-none]
width	location to return width needed to render a cell, or `NULL`.	[out][allow-none]
height	location to return height needed to render a cell, or `NULL`.	[out][allow-none]

---

gtk_cell_renderer_render ()

```
void
gtk_cell_renderer_render (GtkCellRenderer *cell,
                          cairo_t *cr,
                          GtkWidget *widget,
                          const GdkRectangle *background_area,
                          const GdkRectangle *cell_area,
```

```
                                GtkCellRendererState flags);
```

Invokes the virtual render function of the GtkCellRenderer. The three passed-in rectangles are areas in `cr` . Most renderers will draw within `cell_area` ; the xalign, yalign, xpad, and ypad fields of the GtkCellRenderer should be honored with respect to `cell_area` . `background_area` includes the blank space around the cell, and also the area containing the tree expander; so the `background_area` rectangles for all cells tile to cover the entire `window` .

Parameters

cell	a GtkCellRenderer
cr	a cairo context to draw to
widget	the widget owning `window`
background_area	entire cell area (including tree expanders and maybe padding on the sides)
cell_area	area normally rendered by a cell renderer
flags	flags that affect rendering

---

gtk_cell_renderer_activate ()

```
gboolean
gtk_cell_renderer_activate (GtkCellRenderer *cell,
                            GdkEvent *event,
                            GtkWidget *widget,
                            const gchar *path,
                            const GdkRectangle *background_area,
                            const GdkRectangle *cell_area,
                            GtkCellRendererState flags);
```

Passes an activate event to the cell renderer for possible processing. Some cell renderers may use events; for example, GtkCellRendererToggle toggles when it gets a mouse click.

Parameters

cell	a GtkCellRenderer
event	a GdkEvent
widget	widget that received the event
path	widget-dependent string representation of the event location; e.g. for GtkTreeView, a string representation of GtkTreePath
background_area	background area as passed to `gtk_cell_renderer_render()`
cell_area	cell area as passed to `gtk_cell_renderer_render()`
flags	render flags

Returns

`TRUE` if the event was consumed/handled

---

gtk_cell_renderer_start_editing ()

```
GtkCellEditable *
gtk_cell_renderer_start_editing (GtkCellRenderer *cell,
                                 GdkEvent *event,
                                 GtkWidget *widget,
                                 const gchar *path,
                                 const GdkRectangle *background_area,
                                 const GdkRectangle *cell_area,
                                 GtkCellRendererState flags);
```

Starts editing the contents of this `cell` , through a new GtkCellEditable widget created by the GtkCellRendererClass.start_editing virtual function.

Parameters

cell	a GtkCellRenderer	
event	a GdkEvent.	[nullable]
widget	widget that received the event	
path	widget-dependent string representation of the event location; e.g. for GtkTreeView, a string representation of GtkTreePath	
background_area	background area as passed to `gtk_cell_renderer_render()`	
cell_area	cell area as passed to `gtk_cell_renderer_render()`	
flags	render flags	

Returns

A new GtkCellEditable for editing this `cell` , or `NULL` if editing is not possible.

[nullable][transfer none]

---

gtk_cell_renderer_stop_editing ()

```
void
gtk_cell_renderer_stop_editing (GtkCellRenderer *cell,
                                gboolean canceled);
```

Informs the cell renderer that the editing is stopped. If `canceled` is `TRUE`, the cell renderer will emit the "editing-canceled" signal.

This function should be called by cell renderer implementations in response to the "editing-done" signal of GtkCellEditable.

Parameters

cell	A GtkCellRenderer	
canceled	`TRUE` if the editing has been canceled	

Since: 2.6

---

gtk_cell_renderer_get_fixed_size ()

```
void
gtk_cell_renderer_get_fixed_size (GtkCellRenderer *cell,
                                  gint *width,
                                  gint *height);
```

Fills in `width` and `height` with the appropriate size of `cell` .

Parameters

cell	A GtkCellRenderer	
width	location to fill in with the fixed width of the cell, or `NULL`.	[out][allow-none]
height	location to fill in with the fixed height of the cell, or `NULL`.	[out][allow-none]

---

gtk_cell_renderer_set_fixed_size ()

```
void
gtk_cell_renderer_set_fixed_size (GtkCellRenderer *cell,
                                  gint width,
                                  gint height);
```

Sets the renderer size to be explicit, independent of the properties set.

Parameters

cell	A GtkCellRenderer
width	the width of the cell renderer, or -1
height	the height of the cell renderer, or -1

---

gtk_cell_renderer_get_visible ()

```
gboolean
gtk_cell_renderer_get_visible (GtkCellRenderer *cell);
```

Returns the cell renderer’s visibility.

Parameters

cell	A GtkCellRenderer

Returns

`TRUE` if the cell renderer is visible

Since: 2.18

---

gtk_cell_renderer_set_visible ()

```
void
gtk_cell_renderer_set_visible (GtkCellRenderer *cell,
                               gboolean visible);
```

Sets the cell renderer’s visibility.

Parameters

cell	A GtkCellRenderer
visible	the visibility of the cell

Since: 2.18

---

gtk_cell_renderer_get_sensitive ()

```
gboolean
gtk_cell_renderer_get_sensitive (GtkCellRenderer *cell);
```

Returns the cell renderer’s sensitivity.

Parameters

cell	A GtkCellRenderer

Returns

`TRUE` if the cell renderer is sensitive

Since: 2.18

---

gtk_cell_renderer_set_sensitive ()

```
void
gtk_cell_renderer_set_sensitive (GtkCellRenderer *cell,
                                 gboolean sensitive);
```

Sets the cell renderer’s sensitivity.

Parameters

cell	A GtkCellRenderer	
sensitive	the sensitivity of the cell	

Since: 2.18

---

gtk_cell_renderer_get_alignment ()

```
void
gtk_cell_renderer_get_alignment (GtkCellRenderer *cell,
                                 gfloat *xalign,
                                 gfloat *yalign);
```

Fills in `xalign` and `yalign` with the appropriate values of `cell`.

Parameters

cell	A GtkCellRenderer	
xalign	location to fill in with the x alignment of the cell, or `NULL`.	[out][allow-none]
yalign	location to fill in with the y alignment of the cell, or `NULL`.	[out][allow-none]

Since: 2.18

---

gtk_cell_renderer_set_alignment ()

```
void
gtk_cell_renderer_set_alignment (GtkCellRenderer *cell,
                                 gfloat xalign,
                                 gfloat yalign);
```

Sets the renderer's alignment within its available space.

Parameters

cell	A GtkCellRenderer
xalign	the x alignment of the cell renderer
yalign	the y alignment of the cell renderer

Since: 2.18

---

gtk_cell_renderer_get_padding ()

```
void
gtk_cell_renderer_get_padding (GtkCellRenderer *cell,
                               gint *xpad,
                               gint *ypad);
```

Fills in `xpad` and `ypad` with the appropriate values of `cell`.

Parameters

cell	A GtkCellRenderer	
xpad	location to fill in with the x padding of the cell, or `NULL`.	[out][allow-none]
ypad	location to fill in with the y padding of the cell, or `NULL`.	[out][allow-none]

Since: 2.18

---

gtk_cell_renderer_set_padding ()

```
void
gtk_cell_renderer_set_padding (GtkCellRenderer *cell,
                               gint xpad,
                               gint ypad);
```

Sets the renderer's padding.

Parameters

cell	A GtkCellRenderer
xpad	the x padding of the cell renderer
ypad	the y padding of the cell renderer

Since: 2.18

---

gtk_cell_renderer_get_state ()

```
GtkStateFlags
gtk_cell_renderer_get_state (GtkCellRenderer *cell,
                             GtkWidget *widget,
                             GtkCellRendererState cell_state);
```

Translates the cell renderer state to GtkStateFlags, based on the cell renderer and widget sensitivity, and the given GtkCellRendererState.

Parameters

cell	a GtkCellRenderer, or `NULL`.	[nullable]
widget	a GtkWidget, or `NULL`.	[nullable]
cell_state	cell renderer state	

Returns

the widget state flags applying to `cell`

Since: 3.0

---

gtk_cell_renderer_is_activatable ()

```
gboolean
gtk_cell_renderer_is_activatable (GtkCellRenderer *cell);
```

Checks whether the cell renderer can do something when activated.

Parameters

cell	A GtkCellRenderer	

Returns

`TRUE` if the cell renderer can do anything when activated

Since: 3.0

---

gtk_cell_renderer_get_preferred_height ()

```
void
gtk_cell_renderer_get_preferred_height
                                (GtkCellRenderer *cell,
                                 GtkWidget *widget,
                                 gint *minimum_size,
                                 gint *natural_size);
```

Retreives a renderer’s natural size when rendered to `widget` .

Parameters

cell	a GtkCellRenderer instance	
widget	the GtkWidget this cell will be rendering to	
minimum_size	location to store the minimum size, or `NULL`.	[out][allow-none]
natural_size	location to store the natural size, or `NULL`.	[out][allow-none]

Since: 3.0

---

gtk_cell_renderer_get_preferred_height_for_width ()

```
void
gtk_cell_renderer_get_preferred_height_for_width
                                (GtkCellRenderer *cell,
                                 GtkWidget *widget,
                                 gint width,
                                 gint *minimum_height,
                                 gint *natural_height);
```

Retreives a cell renderers’s minimum and natural height if it were rendered to `widget` with the specified `width` .

Parameters

cell	a GtkCellRenderer instance	
widget	the GtkWidget this cell will be rendering to	
width	the size which is available for allocation	
minimum_height	location for storing the minimum size, or `NULL`.	[out][allow-none]
natural_height	location for storing the preferred size, or `NULL`.	[out][allow-none]

Since: 3.0

---

gtk_cell_renderer_get_preferred_size ()

```
void
gtk_cell_renderer_get_preferred_size (GtkCellRenderer *cell,
                                      GtkWidget *widget,
                                      GtkRequisition *minimum_size,
                                      GtkRequisition *natural_size);
```

Retrieves the minimum and natural size of a cell taking into account the widget's preference for height-for-width management.

Parameters

cell	a GtkCellRenderer instance	
widget	the GtkWidget this cell will be rendering to	
minimum_size	location for storing the minimum size, or `NULL`.	[out][allow-none]
natural_size	location for storing the natural size, or `NULL`.	[out][allow-none]

Since: 3.0

---

gtk_cell_renderer_get_preferred_width ()

```
void
gtk_cell_renderer_get_preferred_width (GtkCellRenderer *cell,
                                       GtkWidget *widget,
                                       gint *minimum_size,
                                       gint *natural_size);
```

Retreives a renderer's natural size when rendered to `widget` .

Parameters

cell	a GtkCellRenderer instance	

widget	the GtkWidget this cell will be rendering to	
minimum_size	location to store the minimum size, or NULL.	[out][allow-none]
natural_size	location to store the natural size, or NULL.	[out][allow-none]

Since: 3.0

---

gtk_cell_renderer_get_preferred_width_for_height ()

```
void
gtk_cell_renderer_get_preferred_width_for_height
                                (GtkCellRenderer *cell,
                                 GtkWidget *widget,
                                 gint height,
                                 gint *minimum_width,
                                 gint *natural_width);
```

Retreives a cell renderers's minimum and natural width if it were rendered to `widget` with the specified `height` .

Parameters

cell	a GtkCellRenderer instance	
widget	the GtkWidget this cell will be rendering to	
height	the size which is available for allocation	
minimum_width	location for storing the minimum size, or NULL.	[out][allow-none]
natural_width	location for storing the preferred size, or NULL.	[out][allow-none]

Since: 3.0

---

gtk_cell_renderer_get_request_mode ()

```
GtkSizeRequestMode
gtk_cell_renderer_get_request_mode (GtkCellRenderer *cell);
```

Gets whether the cell renderer prefers a height-for-width layout or a width-for-height layout.

Parameters

cell	a GtkCellRenderer instance

Returns

The GtkSizeRequestMode preferred by this renderer.

Since: 3.0

Types and Values

enum GtkCellRendererState

Tells how a cell is to be rendered.

Members

GTK_CELL_RENDERER_SELECTED	The cell is currently selected, and probably has a selection colored background to render to.
GTK_CELL_RENDERER_PRELIT	The mouse is hovering over the cell.
GTK_CELL_RENDERER_INSENSITIVE	The cell is drawn in an insensitive manner
GTK_CELL_RENDERER_SORTED	The cell is in a sorted row
GTK_CELL_RENDERER_FOCUSED	The cell is in the focus row.
GTK_CELL_RENDERER_EXPANDABLE	The cell is in a row that can be expanded. Since 3.4
GTK_CELL_RENDERER_EXPANDED	The cell is in a row that is expanded. Since 3.4

---

enum GtkCellRendererMode

Identifies how the user can interact with a particular cell.

Members

GTK_CELL_RENDERER_MODE_INERT	The cell is just for display and cannot be interacted with. Note that this doesn’t mean that eg. the row being drawn can’t be selected -- just that a particular element of it cannot be individually modified.
GTK_CELL_RENDERER_MODE_ACTIVATABLE	The cell can be clicked.
GTK_CELL_RENDERER_MODE_EDITABLE	The cell can be edited or otherwise modified.

---

struct GtkCellRenderer

```
struct GtkCellRenderer;
```

---

struct GtkCellRendererClass

```
struct GtkCellRendererClass {
  /* vtable - not signals */
  GtkSizeRequestMode (* get_request_mode)               (GtkCellRenderer      *cell);
  void               (* get_preferred_width)            (GtkCellRenderer      *cell,
                                                         GtkWidget            *widget,
                                                         gint
*minimum_size,
                                                         gint
*natural_size);
  void               (* get_preferred_height_for_width) (GtkCellRenderer      *cell,
                                                         GtkWidget            *widget,
                                                         gint                  width,
                                                         gint
*minimum_height,
                                                         gint
*natural_height);
  void               (* get_preferred_height)           (GtkCellRenderer      *cell,
                                                         GtkWidget            *widget,
                                                         gint
*minimum_size,
                                                         gint
*natural_size);
  void               (* get_preferred_width_for_height) (GtkCellRenderer      *cell,
                                                         GtkWidget            *widget,
                                                         gint                  height,
                                                         gint
*minimum_width,
                                                         gint
*natural_width);
  void               (* get_aligned_area)               (GtkCellRenderer      *cell,
                                                         GtkWidget            *widget,
                                                GtkCellRendererState  flags,
                                                         const GdkRectangle   *cell_area,
                                                         GdkRectangle
*aligned_area);
  void               (* get_size)                       (GtkCellRenderer      *cell,
                                                         GtkWidget            *widget,
                                                         const GdkRectangle   *cell_area,
                                                         gint                 *x_offset,
                                                         gint                 *y_offset,
                                                         gint                 *width,
                                                         gint                 *height);
  void               (* render)                         (GtkCellRenderer      *cell,
                                                         cairo_t              *cr,
                                                         GtkWidget            *widget,
                                                         const GdkRectangle
*background_area,
                                                         const GdkRectangle   *cell_area,
                                                         GtkCellRendererState  flags);
  gboolean           (* activate)                       (GtkCellRenderer      *cell,
                                                         GdkEvent             *event,
                                                         GtkWidget            *widget,
                                                         const gchar          *path,
                                                         const GdkRectangle
```

```
*background_area,
                                                        const GdkRectangle   *cell_area,
                                                        GtkCellRendererState  flags);
  GtkCellEditable *  (* start_editing)                 (GtkCellRenderer      *cell,
                                                        GdkEvent             *event,
                                                        GtkWidget            *widget,
                                                        const gchar          *path,
                                                        const GdkRectangle
*background_area,
                                                        const GdkRectangle   *cell_area,
                                                        GtkCellRendererState  flags);

  /* Signals */
  void (* editing_canceled) (GtkCellRenderer *cell);
  void (* editing_started)  (GtkCellRenderer *cell,
                       GtkCellEditable *editable,
                       const gchar     *path);
};
```

Members

`get_request_mode` ()	Called to gets whether the cell renderer prefers a height-for-width layout or a width-for-height layout.
`get_preferred_width` ()	Called to get a renderer's natural width.
`get_preferred_height_for_width` ()	Called to get a renderer's natural height for width.
`get_preferred_height` ()	Called to get a renderer's natural height.
`get_preferred_width_for_height` ()	Called to get a renderer's natural width for height.
`get_aligned_area` ()	Called to get the aligned area used by `cell` inside `cell_area` .
`get_size` ()	Called to get the width and height needed to render the cell. Deprecated: 3.0.
`render` ()	Called to render the content of the GtkCellRenderer.
`activate` ()	Called to activate the content of the GtkCellRenderer.
`start_editing` ()	Called to initiate editing the content of the GtkCellRenderer.
`editing_canceled` ()	Signal gets emitted when the user cancels the process of editing a cell.
`editing_started` ()	Signal gets emitted when a cell starts to be edited.

Property Details

The `"cell-background"` property

```
  "cell-background"          gchar *
```

Cell background color as a string.

Flags: Write

Default value: NULL

The “cell-background-gdk” property

“cell-background-gdk” GdkColor *

Cell background as a GdkColor

GtkCellRenderer:cell-background-gdk has been deprecated since version 3.4 and should not be used in newly-written code.

Use “cell-background-rgba” instead.

Flags: Read / Write

The “cell-background-rgba” property

“cell-background-rgba” GdkRGBA *

Cell background as a GdkRGBA

Flags: Read / Write

Since: 3.0

The “cell-background-set” property

“cell-background-set” gboolean

Whether the cell background color is set.

Flags: Read / Write

Default value: FALSE

The “editing” property

“editing” gboolean

Whether the cell renderer is currently in editing mode.

Flags: Read

Default value: FALSE

The “height” property

“height” gint

The fixed height.

Flags: Read / Write

Allowed values: >= -1

Default value: -1

The “is-expanded” property

“is-expanded” gboolean

Row is an expander row, and is expanded.

Flags: Read / Write

Default value: FALSE

The “is-expander” property

“is-expander” gboolean

Row has children.

Flags: Read / Write

Default value: FALSE

The “mode” property

“mode” GtkCellRendererMode

Editable mode of the CellRenderer.

Flags: Read / Write

Default value: GTK_CELL_RENDERER_MODE_INERT

The “sensitive” property

“sensitive” gboolean

Display the cell sensitive.

Flags: Read / Write

Default value: TRUE

The “visible” property

“visible” gboolean

Display the cell.

Flags: Read / Write

Default value: TRUE

The “width” property

“width” gint

The fixed width.

Flags: Read / Write

Allowed values: >= -1

Default value: -1

The “xalign” property

“xalign” gfloat

The x-align.

Flags: Read / Write

Allowed values: [0,1]

Default value: 0.5

The “xpad” property

“xpad” guint

The xpad.

Flags: Read / Write

Default value: 0

The “yalign” property

“yalign” gfloat

The y-align.

Flags: Read / Write

Allowed values: [0,1]

Default value: 0.5

The “ypad” property

“ypad” guint

The ypad.

Flags: Read / Write

Default value: 0

Signal Details

The “editing-canceled” signal

```
void
user_function (GtkCellRenderer *renderer,
               gpointer         user_data)
```

This signal gets emitted when the user cancels the process of editing a cell. For example, an editable cell renderer could be written to cancel editing when the user presses Escape.

See also: gtk_cell_renderer_stop_editing().

Parameters

renderer	the object which received the signal
user_data	user data set when the signal handler was connected.

Flags: Run First

Since: 2.4

The “editing-started” signal

```
void
user_function (GtkCellRenderer *renderer,
               GtkCellEditable *editable,
               gchar           *path,
               gpointer         user_data)
```

This signal gets emitted when a cell starts to be edited. The intended use of this signal is to do special setup on `editable` , e.g. adding a GtkEntryCompletion or setting up additional columns in a GtkComboBox.

See gtk_cell_editable_start_editing() for information on the lifecycle of the `editable` and a way to do setup that doesn’t depend on the `renderer` .

Note that GTK+ doesn't guarantee that cell renderers will continue to use the same kind of widget for editing in future releases, therefore you should check the type of `editable` before doing any specific setup, as in the following example:

```
1                                       static void
```

```
2                                                   text_editing_started (GtkCellRenderer *cell,
3                                                                         GtkCellEditable
4                                                   *editable,
5                                                                         const gchar     *path,
6                                                                         gpointer         data)
7                                                   {
8                                                     if (GTK_IS_ENTRY (editable))
9
10                                                      {
11                                                      GtkEntry *entry = GTK_ENTRY (editable);
12
13                                                      // ... create a GtkEntryCompletion
14
15                                                      gtk_entry_set_completion (entry,
16                                                  completion);
                                                      }
                                                    }
```

Parameters

renderer	the object which received the signal
editable	the GtkCellEditable
path	the path identifying the edited cell
user_data	user data set when the signal handler was connected.

Flags: Run First

Since: 2.6

See Also

GtkCellEditable

GtkCellEditable

GtkCellEditable — Interface for widgets that can be used for editing cells

Functions

void	gtk_cell_editable_start_editing ()
void	gtk_cell_editable_editing_done ()
void	gtk_cell_editable_remove_widget ()

Properties

gboolean	editing-canceled	Read / Write

Signals

void	editing-done	Run Last
void	remove-widget	Run Last

Types and Values

	GtkCellEditable
struct	GtkCellEditableIface

Object Hierarchy

```
GInterface
└── GtkCellEditable
```

Prerequisites

GtkCellEditable requires GtkWidget.

Known Implementations

GtkCellEditable is implemented by GtkAppChooserButton, GtkComboBox, GtkComboBoxText, GtkEntry, GtkSearchEntry and GtkSpinButton.

Includes

```
#include <gtk/gtk.h>
```

Description

The GtkCellEditable interface must be implemented for widgets to be usable to edit the contents of a GtkTreeView cell. It provides a way to specify how temporary widgets should be configured for editing, get the new value, etc.

Functions

gtk_cell_editable_start_editing ()

```
void
gtk_cell_editable_start_editing (GtkCellEditable *cell_editable,
                                 GdkEvent *event);
```

Begins editing on a `cell_editable` .

The GtkCellRenderer for the cell creates and returns a GtkCellEditable from `gtk_cell_renderer_start_editing()`, configured for the GtkCellRenderer type.

gtk_cell_editable_start_editing() can then set up `cell_editable` suitably for editing a cell, e.g. making the Esc

key emit “editing-done”.

Note that the `cell_editable` is created on-demand for the current edit; its lifetime is temporary and does not persist across other edits and/or cells.

Parameters

cell_editable	A GtkCellEditable	
event	The GdkEvent that began the editing process, or `NULL` if editing was initiated programmatically.	[nullable]

---

gtk_cell_editable_editing_done ()

```
void
gtk_cell_editable_editing_done (GtkCellEditable *cell_editable);
```

Emits the “editing-done” signal.

Parameters

cell_editable	A GtkCellEditable

---

gtk_cell_editable_remove_widget ()

```
void
gtk_cell_editable_remove_widget (GtkCellEditable *cell_editable);
```

Emits the “remove-widget” signal.

Parameters

cell_editable	A GtkCellEditable

Types and Values

GtkCellEditable

```
typedef struct _GtkCellEditable GtkCellEditable;
```

---

struct GtkCellEditableIface

```
struct GtkCellEditableIface {
  /* signals */
```

```
  void (* editing_done)  (GtkCellEditable *cell_editable);
  void (* remove_widget) (GtkCellEditable *cell_editable);

  /* virtual table */
  void (* start_editing) (GtkCellEditable *cell_editable,
                  GdkEvent        *event);
};
```

Members

`editing_done` ()	Signal is a sign for the cell renderer to update its value from the cell_editable.
`remove_widget` ()	Signal is meant to indicate that the cell is finished editing, and the widget may now be destroyed.
`start_editing` ()	Begins editing on a cell_editable.

Property Details

The `"editing-canceled"` property

```
  "editing-canceled"         gboolean
```

Indicates whether editing on the cell has been canceled.

Flags: Read / Write

Default value: FALSE

Since: 2.20

Signal Details

The `"editing-done"` signal

```
void
user_function (GtkCellEditable *cell_editable,
               gpointer         user_data)
```

This signal is a sign for the cell renderer to update its value from the `cell_editable` .

Implementations of GtkCellEditable are responsible for emitting this signal when they are done editing, e.g. GtkEntry emits this signal when the user presses Enter. Typical things to do in a handler for ::editing-done are to capture the edited value, disconnect the `cell_editable` from signals on the GtkCellRenderer, etc.

gtk_cell_editable_editing_done() is a convenience method for emitting "editing-done".

Parameters

cell_editable	the object on which the signal was emitted

user_data	user data set when the signal handler was connected.

Flags: Run Last

The “remove-widget” signal

```
void
user_function (GtkCellEditable *cell_editable,
               gpointer         user_data)
```

This signal is meant to indicate that the cell is finished editing, and the `cell_editable` widget is being removed and may subsequently be destroyed.

Implementations of GtkCellEditable are responsible for emitting this signal when they are done editing. It must be emitted after the “editing-done” signal, to give the cell renderer a chance to update the cell's value before the widget is removed.

gtk_cell_editable_remove_widget() is a convenience method for emitting “remove-widget”.

Parameters

cell_editable	the object on which the signal was emitted
user_data	user data set when the signal handler was connected.

Flags: Run Last

See Also

GtkCellRenderer

GtkCellRendererAccel

GtkCellRendererAccel — Renders a keyboard accelerator in a cell

Functions

GtkCellRenderer *	gtk_cell_renderer_accel_new ()

Properties

guint	accel-key	Read / Write
GtkCellRendererAccelMode	accel-mode	Read / Write
GdkModifierType	accel-mods	Read / Write
guint	keycode	Read / Write

Signals

void	accel-cleared	Run Last
void	accel-edited	Run Last

Types and Values

struct	GtkCellRendererAccel
enum	GtkCellRendererAccelMode

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkCellRenderer
        └── GtkCellRendererText
            └── GtkCellRendererAccel
```

Includes

```
#include <gtk/gtk.h>
```

Description

GtkCellRendererAccel displays a keyboard accelerator (i.e. a key combination like `Control + a`). If the cell renderer is editable, the accelerator can be changed by simply typing the new combination.

The GtkCellRendererAccel cell renderer was added in GTK+ 2.10.

Functions

gtk_cell_renderer_accel_new ()

```
GtkCellRenderer *
gtk_cell_renderer_accel_new (void);
```

Creates a new GtkCellRendererAccel.

Returns

the new cell renderer

Since: 2.10

Types and Values

struct GtkCellRendererAccel

```
struct GtkCellRendererAccel;
```

---

enum GtkCellRendererAccelMode

Determines if the edited accelerators are GTK+ accelerators. If they are, consumed modifiers are suppressed, only accelerators accepted by GTK+ are allowed, and the accelerators are rendered in the same way as they are in menus.

Members

GTK_CELL_RENDERER_ACCEL_MODE_GTK	GTK+ accelerators mode
GTK_CELL_RENDERER_ACCEL_MODE_OTHER	Other accelerator mode

Property Details

The `"accel-key"` property

`"accel-key"` `guint`

The keyval of the accelerator.

Flags: Read / Write

Allowed values: <= G_MAXINT

Default value: 0

Since: 2.10

The `"accel-mode"` property

`"accel-mode"` `GtkCellRendererAccelMode`

Determines if the edited accelerators are GTK+ accelerators. If they are, consumed modifiers are suppressed, only accelerators accepted by GTK+ are allowed, and the accelerators are rendered in the same way as they are in menus.

Flags: Read / Write

Default value: GTK_CELL_RENDERER_ACCEL_MODE_GTK

Since: 2.10

The “accel-mods” property

```
  “accel-mods”               GdkModifierType
```

The modifier mask of the accelerator.

Flags: Read / Write

Since: 2.10

The “keycode” property

```
  “keycode”                  guint
```

The hardware keycode of the accelerator. Note that the hardware keycode is only relevant if the key does not have a keyval. Normally, the keyboard configuration should assign keyvals to all keys.

Flags: Read / Write

Allowed values: <= G_MAXINT

Default value: 0

Since: 2.10

Signal Details

The “accel-cleared” signal

```
void
user_function (GtkCellRendererAccel *accel,
               gchar                *path_string,
               gpointer              user_data)
```

Gets emitted when the user has removed the accelerator.

Parameters

accel	the object reveiving the signal
path_string	the path identifying the row of the edited cell
user_data	user data set when the signal handler was connected.

Flags: Run Last

Since: 2.10

The “accel-edited” signal

```
void
user_function (GtkCellRendererAccel *accel,
               gchar                *path_string,
```

```
                guint                   accel_key,
                GdkModifierType         accel_mods,
                guint                   hardware_keycode,
                gpointer                user_data)
```

Gets emitted when the user has selected a new accelerator.

Parameters

accel	the object reveiving the signal
path_string	the path identifying the row of the edited cell
accel_key	the new accelerator keyval
accel_mods	the new acclerator modifier mask
hardware_keycode	the keycode of the new accelerator
user_data	user data set when the signal handler was connected.

Flags: Run Last

Since: 2.10

GtkCellRendererCombo

GtkCellRendererCombo — Renders a combobox in a cell

Functions

GtkCellRenderer *	gtk_cell_renderer_combo_new ()

Properties

gboolean	has-entry	Read / Write
GtkTreeModel *	model	Read / Write
gint	text-column	Read / Write

Signals

void	changed	Run Last

Types and Values

struct	GtkCellRendererCombo

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkCellRenderer
```

```
            └── GtkCellRendererText
                └── GtkCellRendererCombo
```

Includes

```
#include <gtk/gtk.h>
```

Description

GtkCellRendererCombo renders text in a cell like GtkCellRendererText from which it is derived. But while GtkCellRendererText offers a simple entry to edit the text, GtkCellRendererCombo offers a GtkComboBox widget to edit the text. The values to display in the combo box are taken from the tree model specified in the “model” property.

The combo cell renderer takes care of adding a text cell renderer to the combo box and sets it to display the column specified by its “text-column” property. Further properties of the combo box can be set in a handler for the “editing-started” signal.

The GtkCellRendererCombo cell renderer was added in GTK+ 2.6.

Functions

gtk_cell_renderer_combo_new ()

```
GtkCellRenderer *
gtk_cell_renderer_combo_new (void);
```

Creates a new GtkCellRendererCombo. Adjust how text is drawn using object properties. Object properties can be set globally (with `g_object_set()`). Also, with GtkTreeViewColumn, you can bind a property to a value in a GtkTreeModel. For example, you can bind the “text” property on the cell renderer to a string value in the model, thus rendering a different string in each row of the GtkTreeView.

Returns

the new cell renderer

Since: 2.6

Types and Values

struct GtkCellRendererCombo

```
struct GtkCellRendererCombo;
```

Property Details

The “has-entry” property

```
  “has-entry”                gboolean
```

If TRUE, the cell renderer will include an entry and allow to enter values other than the ones in the popup list.

Flags: Read / Write

Default value: TRUE

Since: 2.6

The “model” property

```
  “model”                    GtkTreeModel *
```

Holds a tree model containing the possible values for the combo box. Use the text_column property to specify the column holding the values.

Flags: Read / Write

Since: 2.6

The “text-column” property

```
  “text-column”              gint
```

Specifies the model column which holds the possible values for the combo box.

Note that this refers to the model specified in the model property, not the model backing the tree view to which this cell renderer is attached.

GtkCellRendererCombo automatically adds a text cell renderer for this column to its combo box.

Flags: Read / Write

Allowed values: >= -1

Default value: -1

Since: 2.6

Signal Details

The “changed” signal

```
void
user_function (GtkCellRendererCombo *combo,
               gchar                *path_string,
               GtkTreeIter          *new_iter,
               gpointer              user_data)
```

This signal is emitted each time after the user selected an item in the combo box, either by using the mouse or the arrow keys. Contrary to GtkComboBox, GtkCellRendererCombo::changed is not emitted for changes made to a selected item in the entry. The argument new_iter corresponds to the newly selected item in the combo box

and it is relative to the GtkTreeModel set via the model property on GtkCellRendererCombo.

Note that as soon as you change the model displayed in the tree view, the tree view will immediately cease the editing operating. This means that you most probably want to refrain from changing the model until the combo cell renderer emits the edited or editing_canceled signal.

Parameters

combo	the object on which the signal is emitted
path_string	a string of the path identifying the edited cell (relative to the tree view model)
new_iter	the new iter selected in the combo box (relative to the combo box model)
user_data	user data set when the signal handler was connected.

Flags: Run Last

Since: 2.14

GtkCellRendererPixbuf

GtkCellRendererPixbuf — Renders a pixbuf in a cell

Functions

GtkCellRenderer *	gtk_cell_renderer_pixbuf_new ()

Properties

gboolean	follow-state	Read / Write
GIcon *	gicon	Read / Write
gchar *	icon-name	Read / Write
GdkPixbuf *	pixbuf	Read / Write
GdkPixbuf *	pixbuf-expander-closed	Read / Write
GdkPixbuf *	pixbuf-expander-open	Read / Write
gchar *	stock-detail	Read / Write
gchar *	stock-id	Read / Write
guint	stock-size	Read / Write
CairoSurface *	surface	Read / Write

Types and Values

struct	GtkCellRendererPixbuf

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkCellRenderer
        └── GtkCellRendererPixbuf
```

Includes

```
#include <gtk/gtk.h>
```

Description

A GtkCellRendererPixbuf can be used to render an image in a cell. It allows to render either a given GdkPixbuf (set via the “pixbuf” property) or a named icon (set via the “icon-name” property).

To support the tree view, GtkCellRendererPixbuf also supports rendering two alternative pixbufs, when the “is-expander” property is `TRUE`. If the “is-expanded” property is `TRUE` and the “pixbuf-expander-open” property is set to a pixbuf, it renders that pixbuf, if the “is-expanded” property is `FALSE` and the “pixbuf-expander-closed” property is set to a pixbuf, it renders that one.

Functions

gtk_cell_renderer_pixbuf_new ()

```
GtkCellRenderer *
gtk_cell_renderer_pixbuf_new (void);
```

Creates a new GtkCellRendererPixbuf. Adjust rendering parameters using object properties. Object properties can be set globally (with `g_object_set()`). Also, with GtkTreeViewColumn, you can bind a property to a value in a GtkTreeModel. For example, you can bind the “pixbuf” property on the cell renderer to a pixbuf value in the model, thus rendering a different image in each row of the GtkTreeView.

Returns

the new cell renderer

Types and Values

struct GtkCellRendererPixbuf

```
struct GtkCellRendererPixbuf;
```

Property Details

The “follow-state” property

“follow-state” gboolean

Specifies whether the rendered pixbuf should be colorized according to the GtkCellRendererState.

GtkCellRendererPixbuf:follow-state has been deprecated since version 3.16 and should not be used in newly-written code.

Cell renderers always follow state.

Flags: Read / Write

Default value: TRUE

Since: 2.8

The “gicon” property

“gicon” GIcon *

The GIcon representing the icon to display. If the icon theme is changed, the image will be updated automatically.

Flags: Read / Write

Since: 2.14

The “icon-name” property

“icon-name” gchar *

The name of the themed icon to display. This property only has an effect if not overridden by "stock_id" or "pixbuf" properties.

Flags: Read / Write

Default value: NULL

Since: 2.8

The “pixbuf” property

“pixbuf” GdkPixbuf *

The pixbuf to render.

Flags: Read / Write

The “pixbuf-expander-closed” property

“pixbuf-expander-closed” GdkPixbuf *

Pixbuf for closed expander.

Flags: Read / Write

The “pixbuf-expander-open” property

“pixbuf-expander-open” GdkPixbuf *

Pixbuf for open expander.

Flags: Read / Write

The “stock-detail” property

“stock-detail” gchar *

Render detail to pass to the theme engine.

Flags: Read / Write

Default value: NULL

The “stock-id” property

“stock-id” gchar *

The stock ID of the stock icon to render.

GtkCellRendererPixbuf:stock-id has been deprecated since version 3.10 and should not be used in newly-written code.

Use “icon-name” instead.

Flags: Read / Write

Default value: NULL

Since: 2.2

The “stock-size” property

“stock-size” guint

The GtkIconSize value that specifies the size of the rendered icon.

Flags: Read / Write

Default value: 1

Since: 2.2

The “surface” property

“surface”	CairoSurface *

The surface to render.

Flags: Read / Write

Since: 3.10

GtkCellRendererProgress

GtkCellRendererProgress — Renders numbers as progress bars

Functions

GtkCellRenderer *	gtk_cell_renderer_progress_new ()

Properties

gboolean	inverted	Read / Write
gint	pulse	Read / Write
gchar *	text	Read / Write
gfloat	text-xalign	Read / Write
gfloat	text-yalign	Read / Write
gint	value	Read / Write

Types and Values

struct	GtkCellRendererProgress

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkCellRenderer
        └── GtkCellRendererProgress
```

Implemented Interfaces

GtkCellRendererProgress implements GtkOrientable.

Includes

```
#include <gtk/gtk.h>
```

Description

GtkCellRendererProgress renders a numeric value as a progress par in a cell. Additionally, it can display a text on top of the progress bar.

The GtkCellRendererProgress cell renderer was added in GTK+ 2.6.

Functions

gtk_cell_renderer_progress_new ()

```
GtkCellRenderer *
gtk_cell_renderer_progress_new (void);
```

Creates a new GtkCellRendererProgress.

Returns

the new cell renderer

Since: 2.6

Types and Values

struct GtkCellRendererProgress

```
struct GtkCellRendererProgress;
```

Property Details

The “inverted” property

```
  “inverted”                 gboolean
```

Invert the direction in which the progress bar grows.

Flags: Read / Write

Default value: FALSE

The “pulse” property

```
  “pulse”                    gint
```

Setting this to a non-negative value causes the cell renderer to enter "activity mode", where a block bounces back and forth to indicate that some progress is made, without specifying exactly how much.

Each increment of the property causes the block to move by a little bit.

To indicate that the activity has not started yet, set the property to zero. To indicate completion, set the property to `G_MAXINT`.

Flags: Read / Write

Allowed values: >= -1

Default value: -1

Since: 2.12

The `“text”` property

`“text”` `gchar *`

The "text" property determines the label which will be drawn over the progress bar. Setting this property to `NULL` causes the default label to be displayed. Setting this property to an empty string causes no label to be displayed.

Flags: Read / Write

Default value: NULL

Since: 2.6

The `“text-xalign”` property

`“text-xalign”` `gfloat`

The "text-xalign" property controls the horizontal alignment of the text in the progress bar. Valid values range from 0 (left) to 1 (right). Reserved for RTL layouts.

Flags: Read / Write

Allowed values: [0,1]

Default value: 0.5

Since: 2.12

The `“text-yalign”` property

`“text-yalign”` `gfloat`

The "text-yalign" property controls the vertical alignment of the text in the progress bar. Valid values range from 0 (top) to 1 (bottom).

Flags: Read / Write

Allowed values: [0,1]

Default value: 0.5

Since: 2.12

The “value” property

```
  “value”                    gint
```

The "value" property determines the percentage to which the progress bar will be "filled in".

Flags: Read / Write

Allowed values: [0,100]

Default value: 0

Since: 2.6

GtkCellRendererSpin

GtkCellRendererSpin — Renders a spin button in a cell

Functions

GtkCellRenderer *	gtk_cell_renderer_spin_new ()

Properties

GtkAdjustment *	adjustment	Read / Write
gdouble	climb-rate	Read / Write
guint	digits	Read / Write

Types and Values

struct	GtkCellRendererSpin

Object Hierarchy

```
    GObject
    ╰── GInitiallyUnowned
        ╰── GtkCellRenderer
            ╰── GtkCellRendererText
                ╰── GtkCellRendererSpin
```

Includes

```
#include <gtk/gtk.h>
```

Description

GtkCellRendererSpin renders text in a cell like GtkCellRendererText from which it is derived. But while GtkCellRendererText offers a simple entry to edit the text, GtkCellRendererSpin offers a GtkSpinButton

widget. Of course, that means that the text has to be parseable as a floating point number.

The range of the spinbutton is taken from the adjustment property of the cell renderer, which can be set explicitly or mapped to a column in the tree model, like all properties of cell renders. GtkCellRendererSpin also has properties for the “climb-rate” and the number of “digits” to display. Other GtkSpinButton properties can be set in a handler for the “editing-started” signal.

The GtkCellRendererSpin cell renderer was added in GTK+ 2.10.

Functions

gtk_cell_renderer_spin_new ()

```
GtkCellRenderer *
gtk_cell_renderer_spin_new (void);
```

Creates a new GtkCellRendererSpin.

Returns

a new GtkCellRendererSpin

Since: 2.10

Types and Values

struct GtkCellRendererSpin

```
struct GtkCellRendererSpin;
```

Property Details

The “adjustment” property

```
  “adjustment”               GtkAdjustment *
```

The adjustment that holds the value of the spinbutton. This must be non-NULL for the cell renderer to be editable.

Flags: Read / Write

Since: 2.10

The “climb-rate” property

```
  “climb-rate”               gdouble
```

The acceleration rate when you hold down a button.

Flags: Read / Write

Allowed values: >= 0

Default value: 0

Since: 2.10

The “digits” property

“digits” guint

The number of decimal places to display.

Flags: Read / Write

Allowed values: <= 20

Default value: 0

Since: 2.10

See Also

GtkCellRendererText, GtkSpinButton

GtkCellRendererText

GtkCellRendererText — Renders text in a cell

Functions

GtkCellRenderer *	gtk_cell_renderer_text_new ()
void	gtk_cell_renderer_text_set_fixed_height_from_font ()

Properties

gboolean	align-set	Read / Write
PangoAlignment	alignment	Read / Write
PangoAttrList *	attributes	Read / Write
gchar *	background	Write
GdkColor *	background-gdk	Read / Write
GdkRGBA *	background-rgba	Read / Write
gboolean	background-set	Read / Write
gboolean	editable	Read / Write
gboolean	editable-set	Read / Write
PangoEllipsizeMode	ellipsize	Read / Write
gboolean	ellipsize-set	Read / Write

gchar *	family	Read / Write
gboolean	family-set	Read / Write
gchar *	font	Read / Write
PangoFontDescription *	font-desc	Read / Write
gchar *	foreground	Write
GdkColor *	foreground-gdk	Read / Write
GdkRGBA *	foreground-rgba	Read / Write
gboolean	foreground-set	Read / Write
gchar *	language	Read / Write
gboolean	language-set	Read / Write
gchar *	markup	Write
gint	max-width-chars	Read / Write
gchar *	placeholder-text	Read / Write
gint	rise	Read / Write
gboolean	rise-set	Read / Write
gdouble	scale	Read / Write
gboolean	scale-set	Read / Write
gboolean	single-paragraph-mode	Read / Write
gint	size	Read / Write
gdouble	size-points	Read / Write
gboolean	size-set	Read / Write
PangoStretch	stretch	Read / Write
gboolean	stretch-set	Read / Write
gboolean	strikethrough	Read / Write
gboolean	strikethrough-set	Read / Write
PangoStyle	style	Read / Write
gboolean	style-set	Read / Write
gchar *	text	Read / Write
PangoUnderline	underline	Read / Write
gboolean	underline-set	Read / Write
PangoVariant	variant	Read / Write
gboolean	variant-set	Read / Write
gint	weight	Read / Write
gboolean	weight-set	Read / Write
gint	width-chars	Read / Write
PangoWrapMode	wrap-mode	Read / Write
gint	wrap-width	Read / Write

Signals

void	edited	Run Last

Types and Values

struct	GtkCellRendererText

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkCellRenderer
```

```
└── GtkCellRendererText
    ├── GtkCellRendererAccel
    ├── GtkCellRendererCombo
    └── GtkCellRendererSpin
```

Includes

```
#include <gtk/gtk.h>
```

Description

A GtkCellRendererText renders a given text in its cell, using the font, color and style information provided by its properties. The text will be ellipsized if it is too long and the "ellipsize" property allows it.

If the "mode" is `GTK_CELL_RENDERER_MODE_EDITABLE`, the GtkCellRendererText allows to edit its text using an entry.

Functions

gtk_cell_renderer_text_new ()

```
GtkCellRenderer *
gtk_cell_renderer_text_new (void);
```

Creates a new GtkCellRendererText. Adjust how text is drawn using object properties. Object properties can be set globally (with `g_object_set()`). Also, with GtkTreeViewColumn, you can bind a property to a value in a GtkTreeModel. For example, you can bind the "text" property on the cell renderer to a string value in the model, thus rendering a different string in each row of the GtkTreeView

Returns

the new cell renderer

---

gtk_cell_renderer_text_set_fixed_height_from_font ()

```
void
gtk_cell_renderer_text_set_fixed_height_from_font
                                (GtkCellRendererText *renderer,
                                 gint number_of_rows);
```

Sets the height of a renderer to explicitly be determined by the "font" and "y_pad" property set on it. Further changes in these properties do not affect the height, so they must be accompanied by a subsequent call to this function. Using this function is unflexible, and should really only be used if calculating the size of a cell is too slow (ie, a massive number of cells displayed). If `number_of_rows` is -1, then the fixed height is unset, and the height is determined by the properties again.

Parameters

renderer	A GtkCellRendererText
number_of_rows	Number of rows of text each cell renderer is allocated, or -1

Types and Values

struct GtkCellRendererText

```
struct GtkCellRendererText;
```

Property Details

The “align-set” property

```
  “align-set”                gboolean
```

Whether this tag affects the alignment mode.

Flags: Read / Write

Default value: FALSE

The “alignment” property

```
  “alignment”                PangoAlignment
```

Specifies how to align the lines of text with respect to each other.

Note that this property describes how to align the lines of text in case there are several of them. The "xalign" property of GtkCellRenderer, on the other hand, sets the horizontal alignment of the whole text.

Flags: Read / Write

Default value: PANGO_ALIGN_LEFT

Since: 2.10

The “attributes” property

```
  “attributes”               PangoAttrList *
```

A list of style attributes to apply to the text of the renderer.

Flags: Read / Write

The “background” property

“background” gchar *

Background color as a string.

Flags: Write

Default value: NULL

The “background-gdk” property

“background-gdk” GdkColor *

Background color as a GdkColor

GtkCellRendererText:background-gdk has been deprecated since version 3.4 and should not be used in newly-written code.

Use “background-rgba” instead.

Flags: Read / Write

The “background-rgba” property

“background-rgba” GdkRGBA *

Background color as a GdkRGBA

Flags: Read / Write

Since: 3.0

The “background-set” property

“background-set” gboolean

Whether this tag affects the background color.

Flags: Read / Write

Default value: FALSE

The “editable” property

“editable” gboolean

Whether the text can be modified by the user.

Flags: Read / Write

Default value: FALSE

The “editable-set” property

“editable-set” gboolean

Whether this tag affects text editability.

Flags: Read / Write

Default value: FALSE

The “ellipsize” property

“ellipsize” PangoEllipsizeMode

Specifies the preferred place to ellipsize the string, if the cell renderer does not have enough room to display the entire string. Setting it to PANGO_ELLIPSIZE_NONE turns off ellipsizing. See the wrap-width property for another way of making the text fit into a given width.

Flags: Read / Write

Default value: PANGO_ELLIPSIZE_NONE

Since: 2.6

The “ellipsize-set” property

“ellipsize-set” gboolean

Whether this tag affects the ellipsize mode.

Flags: Read / Write

Default value: FALSE

The “family” property

“family” gchar *

Name of the font family, e.g. Sans, Helvetica, Times, Monospace.

Flags: Read / Write

Default value: NULL

The “family-set” property

“family-set” gboolean

Whether this tag affects the font family.

Flags: Read / Write

Default value: FALSE

The “font” property

“font” gchar *

Font description as a string, e.g. "Sans Italic 12".

Flags: Read / Write

Default value: NULL

The “font-desc” property

“font-desc” PangoFontDescription *

Font description as a PangoFontDescription struct.

Flags: Read / Write

The “foreground” property

“foreground” gchar *

Foreground color as a string.

Flags: Write

Default value: NULL

The “foreground-gdk” property

“foreground-gdk” GdkColor *

Foreground color as a GdkColor

`GtkCellRendererText:foreground-gdk` has been deprecated since version 3.4 and should not be used in newly-written code.

Use “foreground-rgba” instead.

Flags: Read / Write

The “foreground-rgba” property

“foreground-rgba” GdkRGBA *

Foreground color as a GdkRGBA

Flags: Read / Write

Since: 3.0

The “foreground-set” property

“foreground-set” gboolean

Whether this tag affects the foreground color.

Flags: Read / Write

Default value: FALSE

The “language” property

“language” gchar *

The language this text is in, as an ISO code. Pango can use this as a hint when rendering the text. If you don't understand this parameter, you probably don't need it.

Flags: Read / Write

Default value: NULL

The “language-set” property

“language-set” gboolean

Whether this tag affects the language the text is rendered as.

Flags: Read / Write

Default value: FALSE

The “markup” property

“markup” gchar *

Marked up text to render.

Flags: Write

Default value: NULL

The “max-width-chars” property

“max-width-chars” gint

The desired maximum width of the cell, in characters. If this property is set to -1, the width will be calculated automatically.

For cell renderers that ellipsize or wrap text; this property controls the maximum reported width of the cell. The

cell should not receive any greater allocation unless it is set to expand in its GtkCellLayout and all of the cell's siblings have received their natural width.

Flags: Read / Write

Allowed values: >= -1

Default value: -1

Since: 3.0

The “placeholder-text” property

“placeholder-text” gchar *

The text that will be displayed in the GtkCellRenderer if “editable” is TRUE and the cell is empty.

Since 3.6

Flags: Read / Write

Default value: NULL

The “rise” property

“rise” gint

Offset of text above the baseline (below the baseline if rise is negative).

Flags: Read / Write

Allowed values: >= -2147483647

Default value: 0

The “rise-set” property

“rise-set” gboolean

Whether this tag affects the rise.

Flags: Read / Write

Default value: FALSE

The “scale” property

“scale” gdouble

Font scaling factor.

Flags: Read / Write

Allowed values: >= 0

Default value: 1

The “scale-set” property

“scale-set” gboolean

Whether this tag scales the font size by a factor.

Flags: Read / Write

Default value: FALSE

The “single-paragraph-mode” property

“single-paragraph-mode” gboolean

Whether to keep all text in a single paragraph.

Flags: Read / Write

Default value: FALSE

The “size” property

“size” gint

Font size.

Flags: Read / Write

Allowed values: >= 0

Default value: 0

The “size-points” property

“size-points” gdouble

Font size in points.

Flags: Read / Write

Allowed values: >= 0

Default value: 0

The “size-set” property

“size-set” gboolean

Whether this tag affects the font size.

Flags: Read / Write

Default value: FALSE

The “stretch” property

“stretch” PangoStretch

Font stretch.

Flags: Read / Write

Default value: PANGO_STRETCH_NORMAL

The “stretch-set” property

“stretch-set” gboolean

Whether this tag affects the font stretch.

Flags: Read / Write

Default value: FALSE

The “strikethrough” property

“strikethrough” gboolean

Whether to strike through the text.

Flags: Read / Write

Default value: FALSE

The “strikethrough-set” property

“strikethrough-set” gboolean

Whether this tag affects strikethrough.

Flags: Read / Write

Default value: FALSE

The “style” property

“style” PangoStyle

Font style.

Flags: Read / Write

Default value: PANGO_STYLE_NORMAL

The “style-set” property

“style-set” gboolean

Whether this tag affects the font style.

Flags: Read / Write

Default value: FALSE

The “text” property

“text” gchar *

Text to render.

Flags: Read / Write

Default value: NULL

The “underline” property

“underline” PangoUnderline

Style of underline for this text.

Flags: Read / Write

Default value: PANGO_UNDERLINE_NONE

The “underline-set” property

“underline-set” gboolean

Whether this tag affects underlining.

Flags: Read / Write

Default value: FALSE

The “variant” property

“variant” PangoVariant

Font variant.

Flags: Read / Write

Default value: PANGO_VARIANT_NORMAL

The “variant-set” property

“variant-set” gboolean

Whether this tag affects the font variant.

Flags: Read / Write

Default value: FALSE

The “weight” property

“weight” gint

Font weight.

Flags: Read / Write

Allowed values: >= 0

Default value: 400

The “weight-set” property

“weight-set” gboolean

Whether this tag affects the font weight.

Flags: Read / Write

Default value: FALSE

The “width-chars” property

“width-chars” gint

The desired width of the cell, in characters. If this property is set to -1, the width will be calculated automatically, otherwise the cell will request either 3 characters or the property value, whichever is greater.

Flags: Read / Write

Allowed values: >= -1

Default value: -1

Since: 2.6

The “wrap-mode” property

“wrap-mode” PangoWrapMode

Specifies how to break the string into multiple lines, if the cell renderer does not have enough room to display the entire string. This property has no effect unless the wrap-width property is set.

Flags: Read / Write

Default value: PANGO_WRAP_CHAR

Since: 2.8

The “wrap-width” property

“wrap-width” gint

Specifies the minimum width at which the text is wrapped. The wrap-mode property can be used to influence at what character positions the line breaks can be placed. Setting wrap-width to -1 turns wrapping off.

Flags: Read / Write

Allowed values: >= -1

Default value: -1

Since: 2.8

Signal Details

The “edited” signal

```
void
user_function (GtkCellRendererText *renderer,
               gchar               *path,
               gchar               *new_text,
               gpointer             user_data)
```

This signal is emitted after `renderer` has been edited.

It is the responsibility of the application to update the model and store `new_text` at the position indicated by `path` .

Parameters

renderer	the object which received the signal
path	the path identifying the edited cell
new_text	the new text
user_data	user data set when the signal

handler was connected.

Flags: Run Last

GtkCellRendererToggle

GtkCellRendererToggle — Renders a toggle button in a cell

Functions

GtkCellRenderer *	gtk_cell_renderer_toggle_new ()
gboolean	gtk_cell_renderer_toggle_get_radio ()
void	gtk_cell_renderer_toggle_set_radio ()
gboolean	gtk_cell_renderer_toggle_get_active ()
void	gtk_cell_renderer_toggle_set_active ()
gboolean	gtk_cell_renderer_toggle_get_activatable ()
void	gtk_cell_renderer_toggle_set_activatable ()

Properties

gboolean	activatable	Read / Write
gboolean	active	Read / Write
gboolean	inconsistent	Read / Write
gint	indicator-size	Read / Write
gboolean	radio	Read / Write

Signals

void	toggled	Run Last

Types and Values

struct	GtkCellRendererToggle

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkCellRenderer
        └── GtkCellRendererToggle
```

Includes

```
#include <gtk/gtk.h>
```

Description

GtkCellRendererToggle renders a toggle button in a cell. The button is drawn as a radio or a checkbutton, depending on the “radio” property. When activated, it emits the “toggled” signal.

Functions

gtk_cell_renderer_toggle_new ()

```
GtkCellRenderer *
gtk_cell_renderer_toggle_new (void);
```

Creates a new GtkCellRendererToggle. Adjust rendering parameters using object properties. Object properties can be set globally (with `g_object_set()`). Also, with GtkTreeViewColumn, you can bind a property to a value in a GtkTreeModel. For example, you can bind the “active” property on the cell renderer to a boolean value in the model, thus causing the check button to reflect the state of the model.

Returns

the new cell renderer

---

gtk_cell_renderer_toggle_get_radio ()

```
gboolean
gtk_cell_renderer_toggle_get_radio (GtkCellRendererToggle *toggle);
```

Returns whether we’re rendering radio toggles rather than checkboxes.

Parameters

toggle	a GtkCellRendererToggle

Returns

`TRUE` if we’re rendering radio toggles rather than checkboxes

---

gtk_cell_renderer_toggle_set_radio ()

```
void
gtk_cell_renderer_toggle_set_radio (GtkCellRendererToggle *toggle,
                                    gboolean radio);
```

If `radio` is `TRUE`, the cell renderer renders a radio toggle (i.e. a toggle in a group of mutually-exclusive toggles). If `FALSE`, it renders a check toggle (a standalone boolean option). This can be set globally for the cell renderer, or changed just before rendering each cell in the model (for GtkTreeView, you set up a per-row setting using GtkTreeViewColumn to associate model columns with cell renderer properties).

Parameters

toggle	a GtkCellRendererToggle
radio	TRUE to make the toggle look like a radio button

---

gtk_cell_renderer_toggle_get_active ()

```
gboolean
gtk_cell_renderer_toggle_get_active (GtkCellRendererToggle *toggle);
```

Returns whether the cell renderer is active. See gtk_cell_renderer_toggle_set_active().

Parameters

toggle	a GtkCellRendererToggle

Returns

TRUE if the cell renderer is active.

---

gtk_cell_renderer_toggle_set_active ()

```
void
gtk_cell_renderer_toggle_set_active (GtkCellRendererToggle *toggle,
                                     gboolean setting);
```

Activates or deactivates a cell renderer.

Parameters

toggle	a GtkCellRendererToggle.
setting	the value to set.

---

gtk_cell_renderer_toggle_get_activatable ()

```
gboolean
gtk_cell_renderer_toggle_get_activatable
                                (GtkCellRendererToggle *toggle);
```

Returns whether the cell renderer is activatable. See gtk_cell_renderer_toggle_set_activatable().

Parameters

toggle	a GtkCellRendererToggle

Returns

TRUE if the cell renderer is activatable.

Since: 2.18

---

gtk_cell_renderer_toggle_set_activatable ()

```
void
gtk_cell_renderer_toggle_set_activatable
                                (GtkCellRendererToggle *toggle,
                                 gboolean setting);
```

Makes the cell renderer activatable.

Parameters

toggle	a GtkCellRendererToggle.
setting	the value to set.

Since: 2.18

Types and Values

struct GtkCellRendererToggle

```
struct GtkCellRendererToggle;
```

Property Details

The “activatable” property

“activatable” gboolean

The toggle button can be activated.

Flags: Read / Write

Default value: TRUE

The “active” property

“active” gboolean

The toggle state of the button.

Flags: Read / Write

Default value: FALSE

The “inconsistent” property

“inconsistent” gboolean

The inconsistent state of the button.

Flags: Read / Write

Default value: FALSE

The “indicator-size” property

“indicator-size” gint

Size of check or radio indicator.

Flags: Read / Write

Allowed values: >= 0

Default value: 0

The “radio” property

“radio” gboolean

Draw the toggle button as a radio button.

Flags: Read / Write

Default value: FALSE

Signal Details

The “toggled” signal

```
void
user_function (GtkCellRendererToggle *cell_renderer,
               gchar                 *path,
               gpointer               user_data)
```

The ::toggled signal is emitted when the cell is toggled.

It is the responsibility of the application to update the model with the correct value to store at `path` . Often this is simply the opposite of the value currently stored at `path` .

Parameters

cell_renderer	the object which received the signal
path	string representation of GtkTreePath describing the event location

user_data	user data set when the signal handler was connected.

Flags: Run Last

GtkCellRendererSpinner

GtkCellRendererSpinner — Renders a spinning animation in a cell

Functions

GtkCellRenderer *	gtk_cell_renderer_spinner_new ()

Properties

gboolean	active	Read / Write
guint	pulse	Read / Write
GtkIconSize	size	Read / Write

Types and Values

struct	GtkCellRendererSpinner

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkCellRenderer
        └── GtkCellRendererSpinner
```

Includes

```
#include <gtk/gtk.h>
```

Description

GtkCellRendererSpinner renders a spinning animation in a cell, very similar to GtkSpinner. It can often be used as an alternative to a GtkCellRendererProgress for displaying indefinite activity, instead of actual progress.

To start the animation in a cell, set the “active” property to `TRUE` and increment the “pulse” property at regular intervals. The usual way to set the cell renderer properties for each cell is to bind them to columns in your tree model using e.g. `gtk_tree_view_column_add_attribute()`.

Functions

gtk_cell_renderer_spinner_new ()

```
GtkCellRenderer *
gtk_cell_renderer_spinner_new (void);
```

Returns a new cell renderer which will show a spinner to indicate activity.

Returns

a new GtkCellRenderer

Since: 2.20

Types and Values

struct GtkCellRendererSpinner

```
struct GtkCellRendererSpinner;
```

Property Details

The “active” property

```
  “active”                   gboolean
```

Whether the spinner is active (ie. shown) in the cell.

Flags: Read / Write

Default value: FALSE

The “pulse” property

```
  “pulse”                    guint
```

Pulse of the spinner. Increment this value to draw the next frame of the spinner animation. Usually, you would update this value in a timeout.

By default, the GtkSpinner widget draws one full cycle of the animation, consisting of 12 frames, in 750 milliseconds.

Flags: Read / Write

Default value: 0

Since: 2.20

The “size” property

“size” GtkIconSize

The GtkIconSize value that specifies the size of the rendered spinner.

Flags: Read / Write

Default value: GTK_ICON_SIZE_MENU

Since: 2.20

See Also

GtkSpinner, GtkCellRendererProgress

GtkListStore

GtkListStore — A list-like data structure that can be used with the GtkTreeView

Functions

GtkListStore *	gtk_list_store_new ()
GtkListStore *	gtk_list_store_newv ()
void	gtk_list_store_set_column_types ()
void	gtk_list_store_set ()
void	gtk_list_store_set_valist ()
void	gtk_list_store_set_value ()
void	gtk_list_store_set_valuesv ()
gboolean	gtk_list_store_remove ()
void	gtk_list_store_insert ()
void	gtk_list_store_insert_before ()
void	gtk_list_store_insert_after ()
void	gtk_list_store_insert_with_values ()
void	gtk_list_store_insert_with_valuesv ()
void	gtk_list_store_prepend ()
void	gtk_list_store_append ()
void	gtk_list_store_clear ()
gboolean	gtk_list_store_iter_is_valid ()
void	gtk_list_store_reorder ()
void	gtk_list_store_swap ()
void	gtk_list_store_move_before ()
void	gtk_list_store_move_after ()

Types and Values

struct	GtkListStore

Object Hierarchy

```
GObject
└── GtkListStore
```

Implemented Interfaces

GtkListStore implements GtkTreeModel, GtkTreeDragSource, GtkTreeDragDest, GtkTreeSortable and GtkBuildable.

Includes

```
#include <gtk/gtk.h>
```

Description

The GtkListStore object is a list model for use with a GtkTreeView widget. It implements the GtkTreeModel interface, and consequentialy, can use all of the methods available there. It also implements the GtkTreeSortable interface so it can be sorted by the view. Finally, it also implements the tree drag and drop interfaces.

The GtkListStore can accept most GObject types as a column type, though it can't accept all custom types. Internally, it will keep a copy of data passed in (such as a string or a boxed pointer). Columns that accept GObjects are handled a little differently. The GtkListStore will keep a reference to the object instead of copying the value. As a result, if the object is modified, it is up to the application writer to call `gtk_tree_model_row_changed()` to emit the "row_changed" signal. This most commonly affects lists with GdkPixbufs stored.

An example for creating a simple list store:

```
enum {
  COLUMN_STRING,
  COLUMN_INT,
  COLUMN_BOOLEAN,
  N_COLUMNS
};

{
  GtkListStore *list_store;
  GtkTreePath *path;
  GtkTreeIter iter;
  gint i;

  list_store = gtk_list_store_new (N_COLUMNS,

G_TYPE_STRING,

G_TYPE_INT,

G_TYPE_BOOLEAN);

  for (i = 0; i < 10; i++)
    {
      gchar *some_data;

      some_data = get_some_data (i);

      // Add a new row to the model
```

```
      gtk_list_store_append (list_store,
&iter);
      gtk_list_store_set (list_store, &iter,
                          COLUMN_STRING,
some_data,
                          COLUMN_INT, i,
                          COLUMN_BOOLEAN,
FALSE,
                          -1);

      // As the store will keep a copy of the
string internally,
      // we free some_data.
      g_free (some_data);
    }

  // Modify a particular row
  path = gtk_tree_path_new_from_string ("4");
  gtk_tree_model_get_iter (GTK_TREE_MODEL
(list_store),
                           &iter,
                           path);
  gtk_tree_path_free (path);
  gtk_list_store_set (list_store, &iter,
                      COLUMN_BOOLEAN, TRUE,
                      -1);
}
```

Performance Considerations

Internally, the GtkListStore was implemented with a linked list with a tail pointer prior to GTK+ 2.6. As a result, it was fast at data insertion and deletion, and not fast at random data access. The GtkListStore sets the GTK_TREE_MODEL_ITERS_PERSIST flag, which means that GtkTreeIters can be cached while the row exists. Thus, if access to a particular row is needed often and your code is expected to run on older versions of GTK+, it is worth keeping the iter around.

Atomic Operations

It is important to note that only the methods `gtk_list_store_insert_with_values()` and `gtk_list_store_insert_with_valuesv()` are atomic, in the sense that the row is being appended to the store and the values filled in in a single operation with regard to GtkTreeModel signaling. In contrast, using e.g. `gtk_list_store_append()` and then `gtk_list_store_set()` will first create a row, which triggers the “row-inserted” signal on GtkListStore. The row, however, is still empty, and any signal handler connecting to “row-inserted” on this particular store should be prepared for the situation that the row might be empty. This is especially important if you are wrapping the GtkListStore inside a GtkTreeModelFilter and are using a GtkTreeModelFilterVisibleFunc. Using any of the non-atomic operations to append rows to the GtkListStore will cause the GtkTreeModelFilterVisibleFunc to be visited with an empty row first; the function must be prepared for that.

GtkListStore as GtkBuildable

The GtkListStore implementation of the GtkBuildable interface allows to specify the model columns with a

<columns> element that may contain multiple <column> elements, each specifying one model column. The “type” attribute specifies the data type for the column.

Additionally, it is possible to specify content for the list store in the UI definition, with the <data> element. It can contain multiple <row> elements, each specifying to content for one row of the list model. Inside a <row>, the <col> elements specify the content for individual cells.

Note that it is probably more common to define your models in the code, and one might consider it a layering violation to specify the content of a list store in a UI definition, data, not presentation, and common wisdom is to separate the two, as far as possible.

An example of a UI Definition fragment for a list store:

```
<object class="GtkListStore">
  <columns>
    <column type="gchararray"/>
    <column type="gchararray"/>
    <column type="gint"/>
  </columns>
  <data>
    <row>
      <col id="0">John</col>
      <col id="1">Doe</col>
      <col id="2">25</col>
    </row>
    <row>
      <col id="0">Johan</col>
      <col id="1">Dahlin</col>
      <col id="2">50</col>
    </row>
  </data>
</object>
```

Functions

gtk_list_store_new ()

```
GtkListStore *
gtk_list_store_new (gint n_columns,
                    ...);
```

Creates a new list store as with `n_columns` columns each of the types passed in. Note that only types derived from standard GObject fundamental types are supported.

As an example, `gtk_list_store_new (3, G_TYPE_INT, G_TYPE_STRING, GDK_TYPE_PIXBUF);` will create a new GtkListStore with three columns, of type int, string and GdkPixbuf respectively.

Parameters

n_columns	number of columns in the list store
...	all GType types for the columns, from first to last

Returns

a new GtkListStore

gtk_list_store_newv ()

```
GtkListStore *
gtk_list_store_newv (gint n_columns,
                     GType *types);
```

Non-vararg creation function. Used primarily by language bindings.

[rename-to gtk_list_store_new]

Parameters

n_columns	number of columns in the list store	
types	an array of GType types for the columns, from first to last.	[array length=n_columns]

Returns

a new GtkListStore.

[transfer full]

---

gtk_list_store_set_column_types ()

```
void
gtk_list_store_set_column_types (GtkListStore *list_store,
                                 gint n_columns,
                                 GType *types);
```

This function is meant primarily for GObjects that inherit from GtkListStore, and should only be used when constructing a new GtkListStore. It will not function after a row has been added, or a method on the GtkTreeModel interface is called.

Parameters

list_store	A GtkListStore	
n_columns	Number of columns for the list store	
types	An array length n of GTypes.	[array length=n_columns]

---

gtk_list_store_set ()

```
void
gtk_list_store_set (GtkListStore *list_store,
                    GtkTreeIter *iter,
                    ...);
```

Sets the value of one or more cells in the row referenced by `iter` . The variable argument list should contain integer column numbers, each column number followed by the value to be set. The list is terminated by a -1. For

example, to set column 0 with type `G_TYPE_STRING` to “Foo”, you would write `gtk_list_store_set (store, iter, 0, "Foo", -1)`.

The value will be referenced by the store if it is a `G_TYPE_OBJECT`, and it will be copied if it is a `G_TYPE_STRING` or `G_TYPE_BOXED`.

Parameters

list_store	a GtkListStore
iter	row iterator
...	pairs of column number and value, terminated with -1

---

gtk_list_store_set_valist ()

```
void
gtk_list_store_set_valist (GtkListStore *list_store,
                           GtkTreeIter *iter,
                           va_list var_args);
```

See `gtk_list_store_set()`; this version takes a va_list for use by language bindings.

Parameters

list_store	A GtkListStore
iter	A valid GtkTreeIter for the row being modified
var_args	va_list of column/value pairs

---

gtk_list_store_set_value ()

```
void
gtk_list_store_set_value (GtkListStore *list_store,
                          GtkTreeIter *iter,
                          gint column,
                          GValue *value);
```

Sets the data in the cell specified by `iter` and `column` . The type of `value` must be convertible to the type of the column.

Parameters

list_store	A GtkListStore
iter	A valid GtkTreeIter for the row being modified
column	column number to modify
value	new value for the cell

---

gtk_list_store_set_valuesv ()

```
void
gtk_list_store_set_valuesv (GtkListStore *list_store,
                            GtkTreeIter *iter,
                            gint *columns,
                            GValue *values,
                            gint n_values);
```

A variant of `gtk_list_store_set_valist()` which takes the columns and values as two arrays, instead of varargs. This function is mainly intended for language-bindings and in case the number of columns to change is not known until run-time.

[rename-to gtk_list_store_set]

Parameters

list_store	A GtkListStore	
iter	A valid GtkTreeIter for the row being modified	
columns	an array of column numbers.	[array length=n_values]
values	an array of GValues.	[array length=n_values]
n_values	the length of the `columns` and `values` arrays	

Since: 2.12

---

gtk_list_store_remove ()

```
gboolean
gtk_list_store_remove (GtkListStore *list_store,
                       GtkTreeIter *iter);
```

Removes the given row from the list store. After being removed, `iter` is set to be the next valid row, or invalidated if it pointed to the last row in `list_store` .

Parameters

list_store	A GtkListStore
iter	A valid GtkTreeIter

Returns

`TRUE` if `iter` is valid, `FALSE` if not.

---

gtk_list_store_insert ()

```
void
gtk_list_store_insert (GtkListStore *list_store,
```

```
                        GtkTreeIter *iter,
                        gint position);
```

Creates a new row at `position` . `iter` will be changed to point to this new row. If `position` is -1 or is larger than the number of rows on the list, then the new row will be empty after this function is called. To fill in values, you need to call `gtk_list_store_set()` or `gtk_list_store_set_value()`.

Parameters

list_store	A GtkListStore	
iter	An unset GtkTreeIter to set to the new row.	[out]
position	position to insert the new row, or -1 for last	

---

gtk_list_store_insert_before ()

```
void
gtk_list_store_insert_before (GtkListStore *list_store,
                              GtkTreeIter *iter,
                              GtkTreeIter *sibling);
```

Inserts a new row before `sibling` . If `sibling` is `NULL`, then the row will be appended to the end of the list. `iter` will be changed to point to this new row. The row will be empty after this function is called. To fill in values, you need to call `gtk_list_store_set()` or `gtk_list_store_set_value()`.

Parameters

list_store	A GtkListStore	
iter	An unset GtkTreeIter to set to the new row.	[out]
sibling	A valid GtkTreeIter, or `NULL`.	[allow-none]

---

gtk_list_store_insert_after ()

```
void
gtk_list_store_insert_after (GtkListStore *list_store,
                             GtkTreeIter *iter,
                             GtkTreeIter *sibling);
```

Inserts a new row after `sibling` . If `sibling` is `NULL`, then the row will be prepended to the beginning of the list. `iter` will be changed to point to this new row. The row will be empty after this function is called. To fill in values, you need to call `gtk_list_store_set()` or `gtk_list_store_set_value()`.

Parameters

list_store	A GtkListStore	
iter	An unset GtkTreeIter to set to the	[out]

	new row.	
sibling	A valid GtkTreeIter, or NULL.	[allow-none]

gtk_list_store_insert_with_values ()

```
void
gtk_list_store_insert_with_values (GtkListStore *list_store,
                                   GtkTreeIter *iter,
                                   gint position,
                                   ...);
```

Creates a new row at position . iter will be changed to point to this new row. If position is -1, or larger than the number of rows in the list, then the new row will be appended to the list. The row will be filled with the values given to this function.

Calling gtk_list_store_insert_with_values (list_store, iter, position...) has the same effect as calling

```
static void
insert_value (GtkListStore *list_store,
              GtkTreeIter  *iter,
              int           position)
{
  gtk_list_store_insert (list_store, iter,
position);
  gtk_list_store_set (list_store,
                      iter
                      // ...
                      );
}
```

with the difference that the former will only emit a row_inserted signal, while the latter will emit row_inserted, row_changed and, if the list store is sorted, rows_reordered. Since emitting the rows_reordered signal repeatedly can affect the performance of the program, gtk_list_store_insert_with_values() should generally be preferred when inserting rows in a sorted list store.

Parameters

list_store	A GtkListStore	
iter	An unset GtkTreeIter to set to the new row, or NULL.	[out][allow-none]
position	position to insert the new row, or -1 to append after existing rows	
...	pairs of column number and value, terminated with -1	

Since: 2.6

gtk_list_store_insert_with_valuesv ()

```
void
gtk_list_store_insert_with_valuesv (GtkListStore *list_store,
                                    GtkTreeIter *iter,
                                    gint position,
                                    gint *columns,
```

```
                                GValue *values,
                                gint n_values);
```

A variant of gtk_list_store_insert_with_values() which takes the columns and values as two arrays, instead of varargs. This function is mainly intended for language-bindings.

Parameters

list_store	A GtkListStore	
iter	An unset GtkTreeIter to set to the new row, or NULL.	[out][allow-none]
position	position to insert the new row, or -1 for last	
columns	an array of column numbers.	[array length=n_values]
values	an array of GValues.	[array length=n_values]
n_values	the length of the columns and values arrays	

Since: 2.6

---

gtk_list_store_prepend ()

```
void
gtk_list_store_prepend (GtkListStore *list_store,
                        GtkTreeIter *iter);
```

Prepends a new row to list_store . iter will be changed to point to this new row. The row will be empty after this function is called. To fill in values, you need to call gtk_list_store_set() or gtk_list_store_set_value().

Parameters

list_store	A GtkListStore	
iter	An unset GtkTreeIter to set to the prepend row.	[out]

---

gtk_list_store_append ()

```
void
gtk_list_store_append (GtkListStore *list_store,
                       GtkTreeIter *iter);
```

Appends a new row to list_store . iter will be changed to point to this new row. The row will be empty after this function is called. To fill in values, you need to call gtk_list_store_set() or gtk_list_store_set_value().

Parameters

list_store	A GtkListStore	
iter	An unset GtkTreeIter to set to the	[out]

appended row.

---

gtk_list_store_clear ()

```
void
gtk_list_store_clear (GtkListStore *list_store);
```

Removes all rows from the list store.

Parameters

list_store	a GtkListStore.

---

gtk_list_store_iter_is_valid ()

```
gboolean
gtk_list_store_iter_is_valid (GtkListStore *list_store,
                              GtkTreeIter *iter);
```

This function is slow. Only use it for debugging and/or testing purposes.

Checks if the given iter is a valid iter for this GtkListStore.

Parameters

list_store	A GtkListStore.
iter	A GtkTreeIter.

Returns

`TRUE` if the iter is valid, `FALSE` if the iter is invalid.

Since: 2.2

---

gtk_list_store_reorder ()

```
void
gtk_list_store_reorder (GtkListStore *store,
                        gint *new_order);
```

Reorders `store` to follow the order indicated by `new_order` . Note that this function only works with unsorted stores.

Parameters

store	A GtkListStore.	
new_order	an array of integers mapping the	[array zero-terminated=1]

	new position of each child to its old position before the re-ordering, i.e. `new_order [newpos] = oldpos`. It must have exactly as many items as the list store's length.

Since: 2.2

---

gtk_list_store_swap ()

```
void
gtk_list_store_swap (GtkListStore *store,
                     GtkTreeIter *a,
                     GtkTreeIter *b);
```

Swaps `a` and `b` in `store` . Note that this function only works with unsorted stores.

Parameters

store	A GtkListStore.	
a	A GtkTreeIter.	
b	Another GtkTreeIter.	

Since: 2.2

---

gtk_list_store_move_before ()

```
void
gtk_list_store_move_before (GtkListStore *store,
                            GtkTreeIter *iter,
                            GtkTreeIter *position);
```

Moves `iter` in `store` to the position before `position` . Note that this function only works with unsorted stores. If `position` is `NULL`, `iter` will be moved to the end of the list.

Parameters

store	A GtkListStore.	
iter	A GtkTreeIter.	
position	A GtkTreeIter, or `NULL`.	[allow-none]

Since: 2.2

---

gtk_list_store_move_after ()

```
void
gtk_list_store_move_after (GtkListStore *store,
                           GtkTreeIter *iter,
                           GtkTreeIter *position);
```

Moves `iter` in `store` to the position after `position` . Note that this function only works with unsorted stores. If `position` is `NULL`, `iter` will be moved to the start of the list.

Parameters

store	A GtkListStore.	
iter	A GtkTreeIter.	
position	A GtkTreeIter or `NULL`.	[allow-none]

Since: 2.2

Types and Values

struct GtkListStore

```
struct GtkListStore;
```

See Also

GtkTreeModel, GtkTreeStore

GtkTreeStore

GtkTreeStore — A tree-like data structure that can be used with the GtkTreeView

Functions

GtkTreeStore *	gtk_tree_store_new ()
GtkTreeStore *	gtk_tree_store_newv ()
void	gtk_tree_store_set_column_types ()
void	gtk_tree_store_set_value ()
void	gtk_tree_store_set ()
void	gtk_tree_store_set_valist ()
void	gtk_tree_store_set_valuesv ()
gboolean	gtk_tree_store_remove ()
void	gtk_tree_store_insert ()
void	gtk_tree_store_insert_before ()
void	gtk_tree_store_insert_after ()
void	gtk_tree_store_insert_with_values ()
void	gtk_tree_store_insert_with_valuesv ()
void	gtk_tree_store_prepend ()
void	gtk_tree_store_append ()
gboolean	gtk_tree_store_is_ancestor ()
gint	gtk_tree_store_iter_depth ()
void	gtk_tree_store_clear ()
gboolean	gtk_tree_store_iter_is_valid ()
void	gtk_tree_store_reorder ()
void	gtk_tree_store_swap ()

void	gtk_tree_store_move_before ()
void	gtk_tree_store_move_after ()

Types and Values

struct	GtkTreeStore

Object Hierarchy

```
GObject
└── GtkTreeStore
```

Implemented Interfaces

GtkTreeStore implements GtkTreeModel, GtkTreeDragSource, GtkTreeDragDest, GtkTreeSortable and GtkBuildable.

Includes

```
#include <gtk/gtk.h>
```

Description

The GtkTreeStore object is a list model for use with a GtkTreeView widget. It implements the GtkTreeModel interface, and consequentially, can use all of the methods available there. It also implements the GtkTreeSortable interface so it can be sorted by the view. Finally, it also implements the tree drag and drop interfaces.

GtkTreeStore as GtkBuildable

The GtkTreeStore implementation of the GtkBuildable interface allows to specify the model columns with a <columns> element that may contain multiple <column> elements, each specifying one model column. The “type” attribute specifies the data type for the column.

An example of a UI Definition fragment for a tree store:

```
<object class="GtkTreeStore">
  <columns>
    <column type="gchararray"/>
    <column type="gchararray"/>
    <column type="gint"/>
  </columns>
</object>
```

Functions

gtk_tree_store_new ()

```
GtkTreeStore *
```

```
gtk_tree_store_new (gint n_columns,
                    ...);
```

Creates a new tree store as with `n_columns` columns each of the types passed in. Note that only types derived from standard GObject fundamental types are supported.

As an example, `gtk_tree_store_new (3, G_TYPE_INT, G_TYPE_STRING, GDK_TYPE_PIXBUF);` will create a new GtkTreeStore with three columns, of type gint, gchararray, and GdkPixbuf respectively.

Parameters

n_columns	number of columns in the tree store
...	all GType types for the columns, from first to last

Returns

a new GtkTreeStore

---

gtk_tree_store_newv ()

```
GtkTreeStore *
gtk_tree_store_newv (gint n_columns,
                     GType *types);
```

Non vararg creation function. Used primarily by language bindings.

[rename-to gtk_tree_store_new]

Parameters

n_columns	number of columns in the tree store	
types	an array of GType types for the columns, from first to last.	[array length=n_columns]

Returns

a new GtkTreeStore.

[transfer full]

---

gtk_tree_store_set_column_types ()

```
void
gtk_tree_store_set_column_types (GtkTreeStore *tree_store,
                                 gint n_columns,
                                 GType *types);
```

This function is meant primarily for GObjects that inherit from GtkTreeStore, and should only be used when constructing a new GtkTreeStore. It will not function after a row has been added, or a method on the GtkTreeModel interface is called.

Parameters

tree_store	A GtkTreeStore	
n_columns	Number of columns for the tree store	
types	An array of GType types, one for each column.	[array length=n_columns]

---

gtk_tree_store_set_value ()

```
void
gtk_tree_store_set_value (GtkTreeStore *tree_store,
                          GtkTreeIter *iter,
                          gint column,
                          GValue *value);
```

Sets the data in the cell specified by `iter` and `column` . The type of `value` must be convertible to the type of the column.

Parameters

tree_store	a GtkTreeStore
iter	A valid GtkTreeIter for the row being modified
column	column number to modify
value	new value for the cell

---

gtk_tree_store_set ()

```
void
gtk_tree_store_set (GtkTreeStore *tree_store,
                    GtkTreeIter *iter,
                    ...);
```

Sets the value of one or more cells in the row referenced by `iter` . The variable argument list should contain integer column numbers, each column number followed by the value to be set. The list is terminated by a -1. For example, to set column 0 with type `G_TYPE_STRING` to “Foo”, you would write `gtk_tree_store_set (store, iter, 0, "Foo", -1)`.

The value will be referenced by the store if it is a `G_TYPE_OBJECT`, and it will be copied if it is a `G_TYPE_STRING` or `G_TYPE_BOXED`.

Parameters

tree_store	A GtkTreeStore
iter	A valid GtkTreeIter for the row being modified
...	pairs of column number and value,

terminated with -1

gtk_tree_store_set_valist ()

```
void
gtk_tree_store_set_valist (GtkTreeStore *tree_store,
                           GtkTreeIter *iter,
                           va_list var_args);
```

See `gtk_tree_store_set()`; this version takes a va_list for use by language bindings.

Parameters

tree_store	A GtkTreeStore
iter	A valid GtkTreeIter for the row being modified
var_args	va_list of column/value pairs

gtk_tree_store_set_valuesv ()

```
void
gtk_tree_store_set_valuesv (GtkTreeStore *tree_store,
                            GtkTreeIter *iter,
                            gint *columns,
                            GValue *values,
                            gint n_values);
```

A variant of `gtk_tree_store_set_valist()` which takes the columns and values as two arrays, instead of varargs. This function is mainly intended for language bindings or in case the number of columns to change is not known until run-time.

[rename-to gtk_tree_store_set]

Parameters

tree_store	A GtkTreeStore	
iter	A valid GtkTreeIter for the row being modified	
columns	an array of column numbers.	[array length=n_values]
values	an array of GValues.	[array length=n_values]
n_values	the length of the `columns` and `values` arrays	

Since: 2.12

gtk_tree_store_remove ()

```
gboolean
gtk_tree_store_remove (GtkTreeStore *tree_store,
```

```
                        GtkTreeIter *iter);
```

Removes `iter` from `tree_store` . After being removed, `iter` is set to the next valid row at that level, or invalidated if it previously pointed to the last one.

Parameters

tree_store	A GtkTreeStore
iter	A valid GtkTreeIter

Returns

`TRUE` if `iter` is still valid, `FALSE` if not.

---

gtk_tree_store_insert ()

```
void
gtk_tree_store_insert (GtkTreeStore *tree_store,
                       GtkTreeIter *iter,
                       GtkTreeIter *parent,
                       gint position);
```

Creates a new row at `position` . If parent is non-`NULL`, then the row will be made a child of `parent` . Otherwise, the row will be created at the toplevel. If `position` is -1 or is larger than the number of rows at that level, then the new row will be inserted to the end of the list. `iter` will be changed to point to this new row. The row will be empty after this function is called. To fill in values, you need to call `gtk_tree_store_set()` or `gtk_tree_store_set_value()`.

Parameters

tree_store	A GtkTreeStore	
iter	An unset GtkTreeIter to set to the new row.	[out]
parent	A valid GtkTreeIter, or `NULL`.	[allow-none]
position	position to insert the new row, or -1 for last	

---

gtk_tree_store_insert_before ()

```
void
gtk_tree_store_insert_before (GtkTreeStore *tree_store,
                              GtkTreeIter *iter,
                              GtkTreeIter *parent,
                              GtkTreeIter *sibling);
```

Inserts a new row before `sibling` . If `sibling` is `NULL`, then the row will be appended to `parent` 's children. If `parent` and `sibling` are `NULL`, then the row will be appended to the toplevel. If both `sibling` and `parent` are set, then `parent` must be the parent of `sibling` . When `sibling` is set, `parent` is optional.

`iter` will be changed to point to this new row. The row will be empty after this function is called. To fill in values, you need to call `gtk_tree_store_set()` or `gtk_tree_store_set_value()`.

Parameters

tree_store	A GtkTreeStore	
iter	An unset GtkTreeIter to set to the new row.	[out]
parent	A valid GtkTreeIter, or NULL.	[allow-none]
sibling	A valid GtkTreeIter, or NULL.	[allow-none]

---

gtk_tree_store_insert_after ()

```
void
gtk_tree_store_insert_after (GtkTreeStore *tree_store,
                             GtkTreeIter *iter,
                             GtkTreeIter *parent,
                             GtkTreeIter *sibling);
```

Inserts a new row after `sibling` . If `sibling` is `NULL`, then the row will be prepended to `parent` 's children. If `parent` and `sibling` are `NULL`, then the row will be prepended to the toplevel. If both `sibling` and `parent` are set, then `parent` must be the parent of `sibling` . When `sibling` is set, `parent` is optional.

`iter` will be changed to point to this new row. The row will be empty after this function is called. To fill in values, you need to call `gtk_tree_store_set()` or `gtk_tree_store_set_value()`.

Parameters

tree_store	A GtkTreeStore	
iter	An unset GtkTreeIter to set to the new row.	[out]
parent	A valid GtkTreeIter, or NULL.	[allow-none]
sibling	A valid GtkTreeIter, or NULL.	[allow-none]

---

gtk_tree_store_insert_with_values ()

```
void
gtk_tree_store_insert_with_values (GtkTreeStore *tree_store,
                                   GtkTreeIter *iter,
                                   GtkTreeIter *parent,
                                   gint position,
                                   ...);
```

Creates a new row at `position` . `iter` will be changed to point to this new row. If `position` is -1, or larger than the number of rows on the list, then the new row will be appended to the list. The row will be filled with the values given to this function.

Calling `gtk_tree_store_insert_with_values (tree_store, iter, position, ...)` has the same effect as calling

```
1                                          gtk_tree_store_insert (tree_store, iter,
2                                          position);
                                           gtk_tree_store_set (tree_store, iter, ...);
```

with the different that the former will only emit a row_inserted signal, while the latter will emit row_inserted,

row_changed and if the tree store is sorted, rows_reordered. Since emitting the rows_reordered signal repeatedly can affect the performance of the program, `gtk_tree_store_insert_with_values()` should generally be preferred when inserting rows in a sorted tree store.

Parameters

tree_store	A GtkTreeStore	
iter	An unset GtkTreeIter to set the new row, or `NULL`.	[out][allow-none]
parent	A valid GtkTreeIter, or `NULL`.	[allow-none]
position	position to insert the new row, or -1 to append after existing rows	
...	pairs of column number and value, terminated with -1	

Since: 2.10

---

gtk_tree_store_insert_with_valuesv ()

```
void
gtk_tree_store_insert_with_valuesv (GtkTreeStore *tree_store,
                                    GtkTreeIter *iter,
                                    GtkTreeIter *parent,
                                    gint position,
                                    gint *columns,
                                    GValue *values,
                                    gint n_values);
```

A variant of `gtk_tree_store_insert_with_values()` which takes the columns and values as two arrays, instead of varargs. This function is mainly intended for language bindings.

[rename-to gtk_tree_store_insert_with_values]

Parameters

tree_store	A GtkTreeStore	
iter	An unset GtkTreeIter to set the new row, or `NULL`.	[out][allow-none]
parent	A valid GtkTreeIter, or `NULL`.	[allow-none]
position	position to insert the new row, or -1 for last	
columns	an array of column numbers.	[array length=n_values]
values	an array of GValues.	[array length=n_values]
n_values	the length of the `columns` and `values` arrays	

Since: 2.10

---

gtk_tree_store_prepend ()

```
void
gtk_tree_store_prepend (GtkTreeStore *tree_store,
                        GtkTreeIter *iter,
                        GtkTreeIter *parent);
```

Prepends a new row to `tree_store` . If `parent` is non-`NULL`, then it will prepend the new row before the first child of `parent` , otherwise it will prepend a row to the top level. `iter` will be changed to point to this new row. The row will be empty after this function is called. To fill in values, you need to call `gtk_tree_store_set()` or `gtk_tree_store_set_value()`.

Parameters

tree_store	A GtkTreeStore	
iter	An unset GtkTreeIter to set to the prepended row.	[out]
parent	A valid GtkTreeIter, or `NULL`.	[allow-none]

---

gtk_tree_store_append ()

```
void
gtk_tree_store_append (GtkTreeStore *tree_store,
                       GtkTreeIter *iter,
                       GtkTreeIter *parent);
```

Appends a new row to `tree_store` . If `parent` is non-`NULL`, then it will append the new row after the last child of `parent` , otherwise it will append a row to the top level. `iter` will be changed to point to this new row. The row will be empty after this function is called. To fill in values, you need to call `gtk_tree_store_set()` or `gtk_tree_store_set_value()`.

Parameters

tree_store	A GtkTreeStore	
iter	An unset GtkTreeIter to set to the appended row.	[out]
parent	A valid GtkTreeIter, or `NULL`.	[allow-none]

---

gtk_tree_store_is_ancestor ()

```
gboolean
gtk_tree_store_is_ancestor (GtkTreeStore *tree_store,
                            GtkTreeIter *iter,
                            GtkTreeIter *descendant);
```

Returns `TRUE` if `iter` is an ancestor of `descendant` . That is, `iter` is the parent (or grandparent or great-grandparent) of `descendant` .

Parameters

tree_store	A GtkTreeStore
iter	A valid GtkTreeIter
descendant	A valid GtkTreeIter

Returns

`TRUE`, if `iter` is an ancestor of `descendant`

---

gtk_tree_store_iter_depth ()

```
gint
gtk_tree_store_iter_depth (GtkTreeStore *tree_store,
                           GtkTreeIter *iter);
```

Returns the depth of `iter` . This will be 0 for anything on the root level, 1 for anything down a level, etc.

Parameters

tree_store	A GtkTreeStore
iter	A valid GtkTreeIter

Returns

The depth of `iter`

---

gtk_tree_store_clear ()

```
void
gtk_tree_store_clear (GtkTreeStore *tree_store);
```

Removes all rows from `tree_store`

Parameters

tree_store	a GtkTreeStore

---

gtk_tree_store_iter_is_valid ()

```
gboolean
gtk_tree_store_iter_is_valid (GtkTreeStore *tree_store,
                              GtkTreeIter *iter);
```

WARNING: This function is slow. Only use it for debugging and/or testing purposes.

Checks if the given iter is a valid iter for this GtkTreeStore.

Parameters

tree_store	A GtkTreeStore.
iter	A GtkTreeIter.

Returns

`TRUE` if the iter is valid, `FALSE` if the iter is invalid.

Since: 2.2

---

gtk_tree_store_reorder ()

```
void
gtk_tree_store_reorder (GtkTreeStore *tree_store,
                        GtkTreeIter *parent,
                        gint *new_order);
```

Reorders the children of `parent` in `tree_store` to follow the order indicated by `new_order` . Note that this function only works with unsorted stores.

[skip]

Parameters

tree_store	A GtkTreeStore	
parent	A GtkTreeIter, or `NULL`.	[nullable]
new_order	an array of integers mapping the new position of each child to its old position before the re-ordering, i.e. `new_order [newpos] = oldpos`.	[array]

Since: 2.2

---

gtk_tree_store_swap ()

```
void
gtk_tree_store_swap (GtkTreeStore *tree_store,
                     GtkTreeIter *a,
                     GtkTreeIter *b);
```

Swaps `a` and `b` in the same level of `tree_store` . Note that this function only works with unsorted stores.

Parameters

tree_store	A GtkTreeStore.
a	A GtkTreeIter.
b	Another GtkTreeIter.

Since: 2.2

---

gtk_tree_store_move_before ()

```
void
gtk_tree_store_move_before (GtkTreeStore *tree_store,
                            GtkTreeIter *iter,
                            GtkTreeIter *position);
```

Moves `iter` in `tree_store` to the position before `position`. `iter` and `position` should be in the same level. Note that this function only works with unsorted stores. If `position` is `NULL`, `iter` will be moved to the end of the level.

Parameters

tree_store	A GtkTreeStore.	
iter	A GtkTreeIter.	
position	A GtkTreeIter or `NULL`.	[allow-none]

Since: 2.2

---

gtk_tree_store_move_after ()

```
void
gtk_tree_store_move_after (GtkTreeStore *tree_store,
                           GtkTreeIter *iter,
                           GtkTreeIter *position);
```

Moves `iter` in `tree_store` to the position after `position`. `iter` and `position` should be in the same level. Note that this function only works with unsorted stores. If `position` is `NULL`, `iter` will be moved to the start of the level.

Parameters

tree_store	A GtkTreeStore.	
iter	A GtkTreeIter.	
position	A GtkTreeIter.	[allow-none]

Since: 2.2

Types and Values

struct GtkTreeStore

```
struct GtkTreeStore;
```

See Also

GtkTreeModel

Menus, Combo Box, Toolbar

GtkComboBox — A widget used to choose from a list of items

GtkComboBoxText — A simple, text-only combo box

GtkMenu — A menu widget

GtkMenuBar — A subclass of GtkMenuShell which holds GtkMenuItem widgets

GtkMenuItem — The widget used for item in menus

GtkRadioMenuItem — A choice from multiple check menu items

GtkCheckMenuItem — A menu item with a check box

GtkSeparatorMenuItem — A separator used in menus

GtkToolShell — Interface for containers containing GtkToolItem widgets

GtkToolbar — Create bars of buttons and other widgets

GtkToolItem — The base class of widgets that can be added to GtkToolShell

GtkToolPalette — A tool palette with categories

GtkToolItemGroup — A sub container used in a tool palette

GtkSeparatorToolItem — A toolbar item that separates groups of other toolbar items

GtkToolButton — A GtkToolItem subclass that displays buttons

GtkMenuToolButton — A GtkToolItem containing a button with an additional dropdown menu

GtkToggleToolButton — A GtkToolItem containing a toggle button

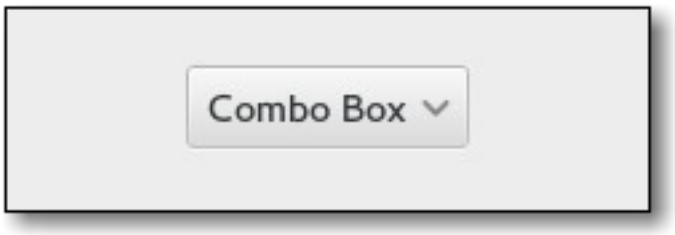
GtkRadioToolButton — A toolbar item that contains a radio button

GtkPopover — Context dependent bubbles

GtkPopoverMenu — Popovers to use as menus

GtkComboBox

GtkComboBox — A widget used to choose from a list of items



Functions

GtkWidget *	gtk_combo_box_new ()
GtkWidget *	gtk_combo_box_new_with_entry ()
GtkWidget *	gtk_combo_box_new_with_model ()
GtkWidget *	gtk_combo_box_new_with_model_and_entry ()
GtkWidget *	gtk_combo_box_new_with_area ()
GtkWidget *	gtk_combo_box_new_with_area_and_entry ()

gint	gtk_combo_box_get_wrap_width ()
void	gtk_combo_box_set_wrap_width ()
gint	gtk_combo_box_get_row_span_column ()
void	gtk_combo_box_set_row_span_column ()
gint	gtk_combo_box_get_column_span_column ()
void	gtk_combo_box_set_column_span_column ()
gint	gtk_combo_box_get_active ()
void	gtk_combo_box_set_active ()
gboolean	gtk_combo_box_get_active_iter ()
void	gtk_combo_box_set_active_iter ()
gint	gtk_combo_box_get_id_column ()
void	gtk_combo_box_set_id_column ()
const gchar *	gtk_combo_box_get_active_id ()
gboolean	gtk_combo_box_set_active_id ()
GtkTreeModel *	gtk_combo_box_get_model ()
void	gtk_combo_box_set_model ()
void	gtk_combo_box_popup_for_device ()
void	gtk_combo_box_popup ()
void	gtk_combo_box_popdown ()
AtkObject *	gtk_combo_box_get_popup_accessible ()
GtkTreeViewRowSeparatorFunc	gtk_combo_box_get_row_separator_func ()
void	gtk_combo_box_set_row_separator_func ()
void	gtk_combo_box_set_add_tearoffs ()
gboolean	gtk_combo_box_get_add_tearoffs ()
void	gtk_combo_box_set_title ()
const gchar *	gtk_combo_box_get_title ()
void	gtk_combo_box_set_focus_on_click ()
gboolean	gtk_combo_box_get_focus_on_click ()
void	gtk_combo_box_set_button_sensitivity ()
GtkSensitivityType	gtk_combo_box_get_button_sensitivity ()
gboolean	gtk_combo_box_get_has_entry ()
void	gtk_combo_box_set_entry_text_column ()
gint	gtk_combo_box_get_entry_text_column ()
void	gtk_combo_box_set_popup_fixed_width ()
gboolean	gtk_combo_box_get_popup_fixed_width ()

Properties

gint	active	Read / Write
gchar *	active-id	Read / Write
gboolean	add-tearoffs	Read / Write
GtkSensitivityType	button-sensitivity	Read / Write
GtkCellArea *	cell-area	Read / Write / Construct Only
gint	column-span-column	Read / Write
gint	entry-text-column	Read / Write
gboolean	has-entry	Read / Write / Construct Only
gboolean	has-frame	Read / Write
gint	id-column	Read / Write
GtkTreeModel *	model	Read / Write
gboolean	popup-fixed-width	Read / Write
gboolean	popup-shown	Read

gint	row-span-column	Read / Write
gchar *	tearoff-title	Read / Write
gint	wrap-width	Read / Write

Style Properties

gboolean	appears-as-list	Read
gfloat	arrow-scaling	Read
gint	arrow-size	Read
GtkShadowType	shadow-type	Read

Signals

void	changed	Run Last
gchar*	format-entry-text	Run Last
void	move-active	Action
gboolean	popdown	Action
void	popup	Action

Types and Values

struct	GtkComboBox
struct	GtkComboBoxClass

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkContainer
            └── GtkBin
                └── GtkComboBox
                    ├── GtkAppChooserButton
                    └── GtkComboBoxText
```

Implemented Interfaces

GtkComboBox implements AtkImplementorIface, GtkBuildable, GtkCellLayout and GtkCellEditable.

Includes

```
#include <gtk/gtk.h>
```

Description

A GtkComboBox is a widget that allows the user to choose from a list of valid choices. The GtkComboBox displays the selected choice. When activated, the GtkComboBox displays a popup which allows the user to make a new choice. The style in which the selected value is displayed, and the style of the popup is determined by the current theme. It may be similar to a Windows-style combo box.

The GtkComboBox uses the model-view pattern; the list of valid choices is specified in the form of a tree model, and the display of the choices can be adapted to the data in the model by using cell renderers, as you would in a tree view. This is possible since GtkComboBox implements the GtkCellLayout interface. The tree model holding the valid choices is not restricted to a flat list, it can be a real tree, and the popup will reflect the tree structure.

To allow the user to enter values not in the model, the “has-entry” property allows the GtkComboBox to contain a GtkEntry. This entry can be accessed by calling `gtk_bin_get_child()` on the combo box.

For a simple list of textual choices, the model-view API of GtkComboBox can be a bit overwhelming. In this case, GtkComboBoxText offers a simple alternative. Both GtkComboBox and GtkComboBoxText can contain an entry.

CSS nodes

```
combobox
├── box.linked
│   └── button.combo
│       └── box
│           ├── cellview
│           └── arrow
└── window.popup
```

A normal combobox contains a box with the .linked class, a button with the .combo class and inside those buttons, there are a cellview and an arrow.

```
combobox
├── box.linked
│   ├── entry.combo
│   └── button.combo
│       └── box
│           └── arrow
└── window.popup
```

A GtkComboBox with an entry has a single CSS node with name combobox. It contains a box with the .linked class. That box contains an entry and a button, both with the .combo class added. The button also contains another node with name arrow.

Functions

gtk_combo_box_new ()

```
GtkWidget *
gtk_combo_box_new (void);
```

Creates a new empty GtkComboBox.

Returns

A new GtkComboBox.

Since: 2.4

---

gtk_combo_box_new_with_entry ()

```
GtkWidget *
gtk_combo_box_new_with_entry (void);
```

Creates a new empty GtkComboBox with an entry.

Returns

A new GtkComboBox.

Since: 2.24

---

gtk_combo_box_new_with_model ()

```
GtkWidget *
gtk_combo_box_new_with_model (GtkTreeModel *model);
```

Creates a new GtkComboBox with the model initialized to `model`.

Parameters

model	A GtkTreeModel.

Returns

A new GtkComboBox.

Since: 2.4

---

gtk_combo_box_new_with_model_and_entry ()

```
GtkWidget *
gtk_combo_box_new_with_model_and_entry
                                (GtkTreeModel *model);
```

Creates a new empty GtkComboBox with an entry and with the model initialized to `model`.

Parameters

model	A GtkTreeModel

Returns

A new GtkComboBox

Since: 2.24

---

gtk_combo_box_new_with_area ()

```
GtkWidget *
gtk_combo_box_new_with_area (GtkCellArea *area);
```

Creates a new empty GtkComboBox using `area` to layout cells.

Parameters

area	the GtkCellArea to use to layout cell renderers

Returns

A new GtkComboBox.

---

gtk_combo_box_new_with_area_and_entry ()

```
GtkWidget *
gtk_combo_box_new_with_area_and_entry (GtkCellArea *area);
```

Creates a new empty GtkComboBox with an entry.

The new combo box will use `area` to layout cells.

Parameters

area	the GtkCellArea to use to layout cell renderers

Returns

A new GtkComboBox.

---

gtk_combo_box_get_wrap_width ()

```
gint
gtk_combo_box_get_wrap_width (GtkComboBox *combo_box);
```

Returns the wrap width which is used to determine the number of columns for the popup menu. If the wrap width is larger than 1, the combo box is in table mode.

Parameters

combo_box	A GtkComboBox

Returns

the wrap width.

Since: 2.6

---

gtk_combo_box_set_wrap_width ()

```
void
gtk_combo_box_set_wrap_width (GtkComboBox *combo_box,
                              gint width);
```

Sets the wrap width of `combo_box` to be `width` . The wrap width is basically the preferred number of columns when you want the popup to be layed out in a table.

Parameters

combo_box	A GtkComboBox
width	Preferred number of columns

Since: 2.4

---

gtk_combo_box_get_row_span_column ()

```
gint
gtk_combo_box_get_row_span_column (GtkComboBox *combo_box);
```

Returns the column with row span information for `combo_box` .

Parameters

combo_box	A GtkComboBox

Returns

the row span column.

Since: 2.6

---

gtk_combo_box_set_row_span_column ()

```
void
gtk_combo_box_set_row_span_column (GtkComboBox *combo_box,
                                   gint row_span);
```

Sets the column with row span information for `combo_box` to be `row_span` . The row span column contains integers which indicate how many rows an item should span.

Parameters

combo_box	A GtkComboBox.
row_span	A column in the model passed

during construction.

Since: 2.4

---

gtk_combo_box_get_column_span_column ()

```
gint
gtk_combo_box_get_column_span_column (GtkComboBox *combo_box);
```

Returns the column with column span information for `combo_box` .

Parameters

combo_box	A GtkComboBox

Returns

the column span column.

Since: 2.6

---

gtk_combo_box_set_column_span_column ()

```
void
gtk_combo_box_set_column_span_column (GtkComboBox *combo_box,
                                      gint column_span);
```

Sets the column with column span information for `combo_box` to be `column_span` . The column span column contains integers which indicate how many columns an item should span.

Parameters

combo_box	A GtkComboBox
column_span	A column in the model passed during construction

Since: 2.4

---

gtk_combo_box_get_active ()

```
gint
gtk_combo_box_get_active (GtkComboBox *combo_box);
```

Returns the index of the currently active item, or -1 if there’s no active item. If the model is a non-flat treemodel, and the active item is not an immediate child of the root of the tree, this function returns `gtk_tree_path_get_indices (path)[0]`, where `path` is the GtkTreePath of the active item.

Parameters

combo_box	A GtkComboBox

Returns

An integer which is the index of the currently active item, or -1 if there's no active item.

Since: 2.4

---

gtk_combo_box_set_active ()

```
void
gtk_combo_box_set_active (GtkComboBox *combo_box,
                          gint index_);
```

Sets the active item of `combo_box` to be the item at `index` .

Parameters

combo_box	A GtkComboBox	
index_	An index in the model passed during construction, or -1 to have no active item	

Since: 2.4

---

gtk_combo_box_get_active_iter ()

```
gboolean
gtk_combo_box_get_active_iter (GtkComboBox *combo_box,
                               GtkTreeIter *iter);
```

Sets `iter` to point to the currently active item, if any item is active. Otherwise, `iter` is left unchanged.

Parameters

combo_box	A GtkComboBox	
iter	A GtkTreeIter.	[out]

Returns

`TRUE` if `iter` was set, `FALSE` otherwise

Since: 2.4

---

gtk_combo_box_set_active_iter ()

```
void
gtk_combo_box_set_active_iter (GtkComboBox *combo_box,
                               GtkTreeIter *iter);
```

Sets the current active item to be the one referenced by `iter` , or unsets the active item if `iter` is `NULL`.

Parameters

combo_box	A GtkComboBox	
iter	The GtkTreeIter, or `NULL`.	[allow-none]

Since: 2.4

---

gtk_combo_box_get_id_column ()

```
gint
gtk_combo_box_get_id_column (GtkComboBox *combo_box);
```

Returns the column which `combo_box` is using to get string IDs for values from.

Parameters

combo_box	A GtkComboBox

Returns

A column in the data source model of `combo_box` .

Since: 3.0

---

gtk_combo_box_set_id_column ()

```
void
gtk_combo_box_set_id_column (GtkComboBox *combo_box,
                             gint id_column);
```

Sets the model column which `combo_box` should use to get string IDs for values from. The column `id_column` in the model of `combo_box` must be of type `G_TYPE_STRING`.

Parameters

combo_box	A GtkComboBox
id_column	A column in `model` to get string IDs for values from

Since: 3.0

---

gtk_combo_box_get_active_id ()

```
const gchar *
gtk_combo_box_get_active_id (GtkComboBox *combo_box);
```

Returns the ID of the active row of `combo_box` . This value is taken from the active row and the column specified by the "id-column" property of `combo_box` (see `gtk_combo_box_set_id_column()`).

The returned value is an interned string which means that you can compare the pointer by value to other interned strings and that you must not free it.

If the "id-column" property of `combo_box` is not set, or if no row is active, or if the active row has a `NULL` ID value, then `NULL` is returned.

Parameters

combo_box	a GtkComboBox

Returns

the ID of the active row, or `NULL`.

[nullable]

Since: 3.0

---

gtk_combo_box_set_active_id ()

```
gboolean
gtk_combo_box_set_active_id (GtkComboBox *combo_box,
                             const gchar *active_id);
```

Changes the active row of `combo_box` to the one that has an ID equal to `active_id` , or unsets the active row if `active_id` is `NULL`. Rows having a `NULL` ID string cannot be made active by this function.

If the "id-column" property of `combo_box` is unset or if no row has the given ID then the function does nothing and returns `FALSE`.

Parameters

combo_box	a GtkComboBox
active_id	the ID of the row to select, or `NULL`. [allow-none]

Returns

`TRUE` if a row with a matching ID was found. If a `NULL` `active_id` was given to unset the active row, the function always returns `TRUE`.

Since: 3.0

---

gtk_combo_box_get_model ()

```
GtkTreeModel *
gtk_combo_box_get_model (GtkComboBox *combo_box);
```

Returns the GtkTreeModel which is acting as data source for `combo_box` .

Parameters

combo_box	A GtkComboBox	

Returns

A GtkTreeModel which was passed during construction.

[transfer none]

Since: 2.4

---

gtk_combo_box_set_model ()

```
void
gtk_combo_box_set_model (GtkComboBox *combo_box,
                         GtkTreeModel *model);
```

Sets the model used by `combo_box` to be `model` . Will unset a previously set model (if applicable). If model is `NULL`, then it will unset the model.

Note that this function does not clear the cell renderers, you have to call `gtk_cell_layout_clear()` yourself if you need to set up different cell renderers for the new model.

Parameters

combo_box	A GtkComboBox	
model	A GtkTreeModel.	[allow-none]

Since: 2.4

---

gtk_combo_box_popup_for_device ()

```
void
gtk_combo_box_popup_for_device (GtkComboBox *combo_box,
                                GdkDevice *device);
```

Pops up the menu or dropdown list of `combo_box` , the popup window will be grabbed so only `device` and its associated pointer/keyboard are the only GdkDevices able to send events to it.

Parameters

combo_box	a GtkComboBox
device	a GdkDevice

Since: 3.0

---

gtk_combo_box_popup ()

```
void
gtk_combo_box_popup (GtkComboBox *combo_box);
```

Pops up the menu or dropdown list of `combo_box` .

This function is mostly intended for use by accessibility technologies; applications should have little use for it.

Before calling this, `combo_box` must be mapped, or nothing will happen.

Parameters

combo_box	a GtkComboBox

Since: 2.4

---

gtk_combo_box_popdown ()

```
void
gtk_combo_box_popdown (GtkComboBox *combo_box);
```

Hides the menu or dropdown list of `combo_box` .

This function is mostly intended for use by accessibility technologies; applications should have little use for it.

Parameters

combo_box	a GtkComboBox

Since: 2.4

---

gtk_combo_box_get_popup_accessible ()

```
AtkObject *
gtk_combo_box_get_popup_accessible (GtkComboBox *combo_box);
```

Gets the accessible object corresponding to the combo box’s popup.

This function is mostly intended for use by accessibility technologies; applications should have little use for it.

Parameters

combo_box	a GtkComboBox

Returns

the accessible object corresponding to the combo box’s popup.

[transfer none]

Since: 2.6

---

gtk_combo_box_get_row_separator_func ()

```
GtkTreeViewRowSeparatorFunc
gtk_combo_box_get_row_separator_func (GtkComboBox *combo_box);
```

Returns the current row separator function.

[skip]

Parameters

combo_box	a GtkComboBox	

Returns

the current row separator function.

Since: 2.6

---

gtk_combo_box_set_row_separator_func ()

```
void
gtk_combo_box_set_row_separator_func (GtkComboBox *combo_box,
                                      GtkTreeViewRowSeparatorFunc func,
                                      gpointer data,
                                      GDestroyNotify destroy);
```

Sets the row separator function, which is used to determine whether a row should be drawn as a separator. If the row separator function is `NULL`, no separators are drawn. This is the default value.

Parameters

combo_box	a GtkComboBox	
func	a GtkTreeViewRowSeparatorFunc	
data	user data to pass to `func` , or `NULL`.	[allow-none]
destroy	destroy notifier for `data` , or `NULL`.	[allow-none]

Since: 2.6

---

gtk_combo_box_set_add_tearoffs ()

```
void
gtk_combo_box_set_add_tearoffs (GtkComboBox *combo_box,
                                gboolean add_tearoffs);
```

`gtk_combo_box_set_add_tearoffs` has been deprecated since version 3.10 and should not be used in newly-

written code.

Sets whether the popup menu should have a tearoff menu item.

Parameters

combo_box	a GtkComboBox
add_tearoffs	TRUE to add tearoff menu items

Since: 2.6

---

gtk_combo_box_get_add_tearoffs ()

```
gboolean
gtk_combo_box_get_add_tearoffs (GtkComboBox *combo_box);
```

`gtk_combo_box_get_add_tearoffs` has been deprecated since version 3.10 and should not be used in newly-written code.

Gets the current value of the :add-tearoffs property.

Parameters

combo_box	a GtkComboBox

Returns

the current value of the :add-tearoffs property.

---

gtk_combo_box_set_title ()

```
void
gtk_combo_box_set_title (GtkComboBox *combo_box,
                         const gchar *title);
```

`gtk_combo_box_set_title` has been deprecated since version 3.10 and should not be used in newly-written code.

Sets the menu’s title in tearoff mode.

Parameters

combo_box	a GtkComboBox
title	a title for the menu in tearoff mode

Since: 2.10

---

gtk_combo_box_get_title ()

```
const gchar *
gtk_combo_box_get_title (GtkComboBox *combo_box);
```

`gtk_combo_box_get_title` has been deprecated since version 3.10 and should not be used in newly-written code.

Gets the current title of the menu in tearoff mode. See `gtk_combo_box_set_add_tearoffs()`.

Parameters

combo_box	a GtkComboBox

Returns

the menu's title in tearoff mode. This is an internal copy of the string which must not be freed.

Since: 2.10

---

gtk_combo_box_set_focus_on_click ()

```
void
gtk_combo_box_set_focus_on_click (GtkComboBox *combo,
                                  gboolean focus_on_click);
```

`gtk_combo_box_set_focus_on_click` has been deprecated since version 3.20 and should not be used in newly-written code.

Use `gtk_widget_set_focus_on_click()` instead

Sets whether the combo box will grab focus when it is clicked with the mouse. Making mouse clicks not grab focus is useful in places like toolbars where you don't want the keyboard focus removed from the main area of the application.

Parameters

combo	a GtkComboBox
focus_on_click	whether the combo box grabs focus when clicked with the mouse

Since: 2.6

---

gtk_combo_box_get_focus_on_click ()

```
gboolean
gtk_combo_box_get_focus_on_click (GtkComboBox *combo);
```

`gtk_combo_box_get_focus_on_click` has been deprecated since version 3.20 and should not be used in newly-written code.

Use `gtk_widget_get_focus_on_click()` instead

Returns whether the combo box grabs focus when it is clicked with the mouse. See `gtk_combo_box_set_focus_on_click()`.

Parameters

combo	a GtkComboBox

Returns

`TRUE` if the combo box grabs focus when it is clicked with the mouse.

Since: 2.6

---

gtk_combo_box_set_button_sensitivity ()

```
void
gtk_combo_box_set_button_sensitivity (GtkComboBox *combo_box,
                                      GtkSensitivityType sensitivity);
```

Sets whether the dropdown button of the combo box should be always sensitive (`GTK_SENSITIVITY_ON`), never sensitive (`GTK_SENSITIVITY_OFF`) or only if there is at least one item to display (`GTK_SENSITIVITY_AUTO`).

Parameters

combo_box	a GtkComboBox
sensitivity	specify the sensitivity of the dropdown button

Since: 2.14

---

gtk_combo_box_get_button_sensitivity ()

```
GtkSensitivityType
gtk_combo_box_get_button_sensitivity (GtkComboBox *combo_box);
```

Returns whether the combo box sets the dropdown button sensitive or not when there are no items in the model.

Parameters

combo_box	a GtkComboBox

Returns

`GTK_SENSITIVITY_ON` if the dropdown button is sensitive when the model is empty, `GTK_SENSITIVITY_OFF` if the button is always insensitive or `GTK_SENSITIVITY_AUTO` if it is only sensitive as long as the model has one item to be selected.

Since: 2.14

---

gtk_combo_box_get_has_entry ()

```
gboolean
gtk_combo_box_get_has_entry (GtkComboBox *combo_box);
```

Returns whether the combo box has an entry.

Parameters

combo_box	a GtkComboBox

Returns

whether there is an entry in `combo_box` .

Since: 2.24

---

gtk_combo_box_set_entry_text_column ()

```
void
gtk_combo_box_set_entry_text_column (GtkComboBox *combo_box,
                                     gint text_column);
```

Sets the model column which `combo_box` should use to get strings from to be `text_column` . The column `text_column` in the model of `combo_box` must be of type `G_TYPE_STRING`.

This is only relevant if `combo_box` has been created with "has-entry" as `TRUE`.

Parameters

combo_box	A GtkComboBox
text_column	A column in `model` to get the strings from for the internal entry

Since: 2.24

---

gtk_combo_box_get_entry_text_column ()

```
gint
gtk_combo_box_get_entry_text_column (GtkComboBox *combo_box);
```

Returns the column which `combo_box` is using to get the strings from to display in the internal entry.

Parameters

combo_box	A GtkComboBox.

Returns

A column in the data source model of `combo_box` .

Since: 2.24

---

gtk_combo_box_set_popup_fixed_width ()

```
void
gtk_combo_box_set_popup_fixed_width (GtkComboBox *combo_box,
                                     gboolean fixed);
```

Specifies whether the popup's width should be a fixed width matching the allocated width of the combo box.

Parameters

combo_box	a GtkComboBox
fixed	whether to use a fixed popup width

Since: 3.0

---

gtk_combo_box_get_popup_fixed_width ()

```
gboolean
gtk_combo_box_get_popup_fixed_width (GtkComboBox *combo_box);
```

Gets whether the popup uses a fixed width matching the allocated width of the combo box.

Parameters

combo_box	a GtkComboBox

Returns

`TRUE` if the popup uses a fixed width

Since: 3.0

Types and Values

struct GtkComboBox

```
struct GtkComboBox;
```

---

struct GtkComboBoxClass

```
struct GtkComboBoxClass {
  GtkBinClass parent_class;


  /* signals */
  void     (* changed)           (GtkComboBox *combo_box);
  gchar   *(* format_entry_text) (GtkComboBox *combo_box,
                                  const gchar *path);
};
```

Members

`changed` ()	Signal is emitted when the active item is changed.
`format_entry_text` ()	Signal which allows you to change how the text displayed in a combo box's entry is displayed.

Property Details

The `"active"` property

```
  "active"                   gint
```

The item which is currently active. If the model is a non-flat treemodel, and the active item is not an immediate child of the root of the tree, this property has the value `gtk_tree_path_get_indices (path)[0]`, where `path` is the GtkTreePath of the active item.

Flags: Read / Write

Allowed values: >= -1

Default value: -1

Since: 2.4

The `"active-id"` property

```
  "active-id"                gchar *
```

The value of the ID column of the active row.

Flags: Read / Write

Default value: NULL

Since: 3.0

The “add-tearoffs” property

“add-tearoffs” gboolean

The add-tearoffs property controls whether generated menus have tearoff menu items.

Note that this only affects menu style combo boxes.

GtkComboBox:add-tearoffs has been deprecated since version 3.10 and should not be used in newly-written code.

Flags: Read / Write

Default value: FALSE

Since: 2.6

The “button-sensitivity” property

“button-sensitivity” GtkSensitivityType

Whether the dropdown button is sensitive when the model is empty.

Flags: Read / Write

Default value: GTK_SENSITIVITY_AUTO

Since: 2.14

The “cell-area” property

“cell-area” GtkCellArea *

The GtkCellArea used to layout cell renderers for this combo box.

If no area is specified when creating the combo box with gtk_combo_box_new_with_area() a horizontally oriented GtkCellAreaBox will be used.

Flags: Read / Write / Construct Only

Since: 3.0

The “column-span-column” property

“column-span-column” gint

If this is set to a non-negative value, it must be the index of a column of type G_TYPE_INT in the model. The value in that column for each item will determine how many columns that item will span in the popup. Therefore, values in this column must be greater than zero, and the sum of an item’s column position + span should not exceed “wrap-width”.

Flags: Read / Write

Allowed values: >= -1

Default value: -1

Since: 2.4

The “entry-text-column” property

“entry-text-column” gint

The column in the combo box's model to associate with strings from the entry if the combo was created with “has-entry” = TRUE.

Flags: Read / Write

Allowed values: >= -1

Default value: -1

Since: 2.24

The “has-entry” property

“has-entry” gboolean

Whether the combo box has an entry.

Flags: Read / Write / Construct Only

Default value: FALSE

Since: 2.24

The “has-frame” property

“has-frame” gboolean

The has-frame property controls whether a frame is drawn around the entry.

Flags: Read / Write

Default value: TRUE

Since: 2.6

The “id-column” property

“id-column” gint

The column in the combo box's model that provides string IDs for the values in the model, if != -1.

Flags: Read / Write

Allowed values: >= -1

Default value: -1

Since: 3.0

The “model” property

“model” GtkTreeModel *

The model from which the combo box takes the values shown in the list.

Flags: Read / Write

Since: 2.4

The “popup-fixed-width” property

“popup-fixed-width” gboolean

Whether the popup's width should be a fixed width matching the allocated width of the combo box.

Flags: Read / Write

Default value: TRUE

Since: 3.0

The “popup-shown” property

“popup-shown” gboolean

Whether the combo boxes dropdown is popped up. Note that this property is mainly useful, because it allows you to connect to notify::popup-shown.

Flags: Read

Default value: FALSE

Since: 2.10

The “row-span-column” property

“row-span-column” gint

If this is set to a non-negative value, it must be the index of a column of type G_TYPE_INT in the model. The value in that column for each item will determine how many rows that item will span in the popup. Therefore, values in this column must be greater than zero.

Flags: Read / Write

Allowed values: >= -1

Default value: -1

Since: 2.4

The “tearoff-title” property

“tearoff-title” gchar *

A title that may be displayed by the window manager when the popup is torn-off.

GtkComboBox:tearoff-title has been deprecated since version 3.10 and should not be used in newly-written code.

Flags: Read / Write

Default value: NULL

Since: 2.10

The “wrap-width” property

“wrap-width” gint

If wrap-width is set to a positive value, items in the popup will be laid out along multiple columns, starting a new row on reaching the wrap width.

Flags: Read / Write

Allowed values: >= 0

Default value: 0

Since: 2.4

Style Property Details

The “appears-as-list” style property

“appears-as-list” gboolean

Whether dropdowns should look like lists rather than menus.

Flags: Read

Default value: FALSE

The “arrow-scaling” style property

“arrow-scaling” gfloat

Sets the amount of space used up by the combobox arrow, proportional to the font size.

GtkComboBox:arrow-scaling has been deprecated since version 3.20 and should not be used in newly-written code.

use the standard min-width/min-height CSS properties on the arrow node; the value of this style property is ignored.

Flags: Read

Allowed values: [0,2]

Default value: 1

The “arrow-size” style property

“arrow-size” gint

Sets the minimum size of the arrow in the combo box. Note that the arrow size is coupled to the font size, so in case a larger font is used, the arrow will be larger than set by arrow size.

GtkComboBox:arrow-size has been deprecated since version 3.20 and should not be used in newly-written code.

use the standard min-width/min-height CSS properties on the arrow node; the value of this style property is ignored.

Flags: Read

Allowed values: >= 0

Default value: 15

Since: 2.12

The “shadow-type” style property

“shadow-type” GtkShadowType

Which kind of shadow to draw around the combo box.

GtkComboBox:shadow-type has been deprecated since version 3.20 and should not be used in newly-written code.

use CSS styling to change the appearance of the combobox frame; the value of this style property is ignored.

Flags: Read

Default value: GTK_SHADOW_NONE

Since: 2.12

Signal Details

The “changed” signal

```
void
user_function (GtkComboBox *widget,
               gpointer     user_data)
```

The changed signal is emitted when the active item is changed. The can be due to the user selecting a different

item from the list, or due to a call to `gtk_combo_box_set_active_iter()`. It will also be emitted while typing into the entry of a combo box with an entry.

Parameters

widget	the object which received the signal
user_data	user data set when the signal handler was connected.

Flags: Run Last

Since: 2.4

The `"format-entry-text"` signal

```
gchar*
user_function (GtkComboBox *combo,
               gchar       *path,
               gpointer     user_data)
```

For combo boxes that are created with an entry (See GtkComboBox:has-entry).

A signal which allows you to change how the text displayed in a combo box's entry is displayed.

Connect a signal handler which returns an allocated string representing `path` . That string will then be used to set the text in the combo box's entry. The default signal handler uses the text from the GtkComboBox::entry-text-column model column.

Here's an example signal handler which fetches data from the model and displays it in the entry.

```
static gchar*
format_entry_text_callback (GtkComboBox
*combo,
                            const gchar
*path,
                            gpointer
user_data)
{
  GtkTreeIter iter;
  GtkTreeModel model;
  gdouble      value;

  model = gtk_combo_box_get_model (combo);

  gtk_tree_model_get_iter_from_string (model,
&iter, path);
  gtk_tree_model_get (model, &iter,

THE_DOUBLE_VALUE_COLUMN, &value,
                      -1);

  return g_strdup_printf ("%g", value);
}
```

Parameters

combo	the object which received the signal
path	the GtkTreePath string from the

	combo box's current model to format text for
user_data	user data set when the signal handler was connected.

Returns

a newly allocated string representing `path` for the current GtkComboBox model.

[transfer full]

Flags: Run Last

Since: 3.4

The `"move-active"` signal

```
void
user_function (GtkComboBox  *widget,
               GtkScrollType scroll_type,
               gpointer      user_data)
```

The ::move-active signal is a keybinding signal which gets emitted to move the active selection.

Parameters

widget	the object that received the signal
scroll_type	a GtkScrollType
user_data	user data set when the signal handler was connected.

Flags: Action

Since: 2.12

The `"popdown"` signal

```
gboolean
user_function (GtkComboBox *button,
               gpointer     user_data)
```

The ::popdown signal is a keybinding signal which gets emitted to popdown the combo box list.

The default bindings for this signal are Alt+Up and Escape.

Parameters

button	the object which received the signal
user_data	user data set when the signal handler was connected.

Flags: Action

Since: 2.12

The “popup” signal

```
void
user_function (GtkComboBox *widget,
               gpointer     user_data)
```

The ::popup signal is a keybinding signal which gets emitted to popup the combo box list.

The default binding for this signal is Alt+Down.

Parameters

widget	the object that received the signal
user_data	user data set when the signal handler was connected.

Flags: Action

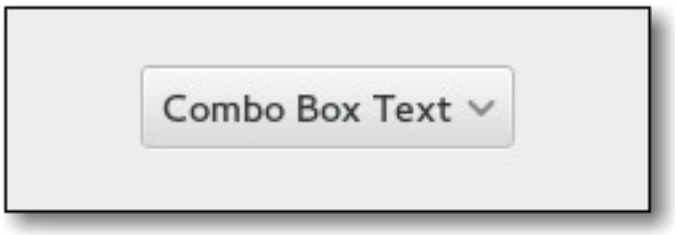
Since: 2.12

See Also

GtkComboBoxText, GtkTreeModel, GtkCellRenderer

GtkComboBoxText

GtkComboBoxText — A simple, text-only combo box



Functions

GtkWidget *	gtk_combo_box_text_new ()
GtkWidget *	gtk_combo_box_text_new_with_entry ()
void	gtk_combo_box_text_append ()
void	gtk_combo_box_text_prepend ()
void	gtk_combo_box_text_insert ()
void	gtk_combo_box_text_append_text ()
void	gtk_combo_box_text_prepend_text ()
void	gtk_combo_box_text_insert_text ()
void	gtk_combo_box_text_remove ()
void	gtk_combo_box_text_remove_all ()
gchar *	gtk_combo_box_text_get_active_text ()

Types and Values

struct	GtkComboBoxText

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkContainer
            └── GtkBin
                └── GtkComboBox
                    └── GtkComboBoxText
```

Implemented Interfaces

GtkComboBoxText implements AtkImplementorIface, GtkBuildable, GtkCellLayout and GtkCellEditable.

Includes

```
#include <gtk/gtk.h>
```

Description

A GtkComboBoxText is a simple variant of GtkComboBox that hides the model-view complexity for simple text-only use cases.

To create a GtkComboBoxText, use `gtk_combo_box_text_new()` or `gtk_combo_box_text_new_with_entry()`.

You can add items to a GtkComboBoxText with `gtk_combo_box_text_append_text()`, `gtk_combo_box_text_insert_text()` or `gtk_combo_box_text_prepend_text()` and remove options with `gtk_combo_box_text_remove()`.

If the GtkComboBoxText contains an entry (via the “has-entry” property), its contents can be retrieved using `gtk_combo_box_text_get_active_text()`. The entry itself can be accessed by calling `gtk_bin_get_child()` on the combo box.

You should not call `gtk_combo_box_set_model()` or attempt to pack more cells into this combo box via its GtkCellLayout interface.

GtkComboBoxText as GtkBuildable

The GtkComboBoxText implementation of the GtkBuildable interface supports adding items directly using the <items> element and specifying <item> elements for each item. Each <item> element can specify the “id” corresponding to the appended text and also supports the regular translation attributes “translatable”, “context” and “comments”.

Here is a UI definition fragment specifying GtkComboBoxText items:

```
1                                                  <object class="GtkComboBoxText">
2                                                    <items>
3                                                      <item translatable="yes"
```

```
4                                                   id="factory">Factory</item>
5                                                       <item translatable="yes" id="home">Home</
6                                                   item>
7                                                       <item translatable="yes"
                                                    id="subway">Subway</item>
                                                      </items>
                                                    </object>
```

CSS nodes

```
1                                                   combobox
2                                                   └── box.linked
3                                                       ├── entry.combo
4                                                       ├── button.combo
5                                                       └── window.popup
```

GtkComboBoxText has a single CSS node with name combobox. It adds the style class .combo to the main CSS nodes of its entry and button children, and the .linked class to the node of its internal box.

Functions

gtk_combo_box_text_new ()

```
GtkWidget *
gtk_combo_box_text_new (void);
```

Creates a new GtkComboBoxText, which is a GtkComboBox just displaying strings.

Returns

A new GtkComboBoxText

Since: 2.24

---

gtk_combo_box_text_new_with_entry ()

```
GtkWidget *
gtk_combo_box_text_new_with_entry (void);
```

Creates a new GtkComboBoxText, which is a GtkComboBox just displaying strings. The combo box created by this function has an entry.

Returns

a new GtkComboBoxText

Since: 2.24

---

gtk_combo_box_text_append ()

```
void
gtk_combo_box_text_append (GtkComboBoxText *combo_box,
                           const gchar *id,
                           const gchar *text);
```

Appends `text` to the list of strings stored in `combo_box` . If `id` is non-`NULL` then it is used as the ID of the row.

This is the same as calling `gtk_combo_box_text_insert()` with a position of -1.

Parameters

combo_box	A GtkComboBoxText	
id	a string ID for this value, or `NULL`.	[allow-none]
text	A string	

Since: 2.24

---

gtk_combo_box_text_prepend ()

```
void
gtk_combo_box_text_prepend (GtkComboBoxText *combo_box,
                            const gchar *id,
                            const gchar *text);
```

Prepends `text` to the list of strings stored in `combo_box` . If `id` is non-`NULL` then it is used as the ID of the row.

This is the same as calling `gtk_combo_box_text_insert()` with a position of 0.

Parameters

combo_box	A GtkComboBox	
id	a string ID for this value, or `NULL`.	[allow-none]
text	a string	

Since: 2.24

---

gtk_combo_box_text_insert ()

```
void
gtk_combo_box_text_insert (GtkComboBoxText *combo_box,
                           gint position,
                           const gchar *id,
                           const gchar *text);
```

Inserts `text` at `position` in the list of strings stored in `combo_box` . If `id` is non-`NULL` then it is used as the ID of the row. See “id-column”.

If `position` is negative then `text` is appended.

Parameters

combo_box	A GtkComboBoxText	
position	An index to insert `text`	
id	a string ID for this value, or `NULL`.	[allow-none]
text	A string to display	

Since: 3.0

---

gtk_combo_box_text_append_text ()

```
void
gtk_combo_box_text_append_text (GtkComboBoxText *combo_box,
                                const gchar *text);
```

Appends `text` to the list of strings stored in `combo_box` .

This is the same as calling `gtk_combo_box_text_insert_text()` with a position of -1.

Parameters

combo_box	A GtkComboBoxText
text	A string

Since: 2.24

---

gtk_combo_box_text_prepend_text ()

```
void
gtk_combo_box_text_prepend_text (GtkComboBoxText *combo_box,
                                 const gchar *text);
```

Prepends `text` to the list of strings stored in `combo_box` .

This is the same as calling `gtk_combo_box_text_insert_text()` with a position of 0.

Parameters

combo_box	A GtkComboBox
text	A string

Since: 2.24

---

gtk_combo_box_text_insert_text ()

```
void
gtk_combo_box_text_insert_text (GtkComboBoxText *combo_box,
                                gint position,
                                const gchar *text);
```

Inserts `text` at `position` in the list of strings stored in `combo_box` .

If `position` is negative then `text` is appended.

This is the same as calling gtk_combo_box_text_insert() with a NULL ID string.

Parameters

combo_box	A GtkComboBoxText
position	An index to insert text
text	A string

Since: 2.24

---

gtk_combo_box_text_remove ()

```
void
gtk_combo_box_text_remove (GtkComboBoxText *combo_box,
                           gint position);
```

Removes the string at position from combo_box .

Parameters

combo_box	A GtkComboBox
position	Index of the item to remove

Since: 2.24

---

gtk_combo_box_text_remove_all ()

```
void
gtk_combo_box_text_remove_all (GtkComboBoxText *combo_box);
```

Removes all the text entries from the combo box.

Parameters

combo_box	A GtkComboBoxText

Since: 3.0

---

gtk_combo_box_text_get_active_text ()

```
gchar *
gtk_combo_box_text_get_active_text (GtkComboBoxText *combo_box);
```

Returns the currently active string in combo_box , or NULL if none is selected. If combo_box contains an entry, this function will return its contents (which will not necessarily be an item from the list).

Parameters

combo_box	A GtkComboBoxText

Returns

a newly allocated string containing the currently active text. Must be freed with `g_free()`.

[transfer full]

Since: 2.24

Types and Values

struct GtkComboBoxText

```
struct GtkComboBoxText;
```

See Also

GtkComboBox

GtkMenu

GtkMenu — A menu widget

Functions

GtkWidget *	gtk_menu_new ()
GtkWidget *	gtk_menu_new_from_model ()
void	gtk_menu_set_screen ()
void	gtk_menu_reorder_child ()
void	gtk_menu_attach ()
void	gtk_menu_popup_at_rect ()
void	gtk_menu_popup_at_widget ()
void	gtk_menu_popup_at_pointer ()
void	gtk_menu_popup_for_device ()
void	gtk_menu_popup ()
void	gtk_menu_set_accel_group ()
GtkAccelGroup *	gtk_menu_get_accel_group ()
void	gtk_menu_set_accel_path ()
const gchar *	gtk_menu_get_accel_path ()
void	gtk_menu_set_title ()
const gchar *	gtk_menu_get_title ()
void	gtk_menu_set_monitor ()
gint	gtk_menu_get_monitor ()
void	gtk_menu_place_on_monitor ()
gboolean	gtk_menu_get_tearoff_state ()
void	gtk_menu_set_reserve_toggle_size ()
gboolean	gtk_menu_get_reserve_toggle_size ()

void	gtk_menu_popdown ()
void	gtk_menu_reposition ()
GtkWidget *	gtk_menu_get_active ()
void	gtk_menu_set_active ()
void	gtk_menu_set_tearoff_state ()
void	gtk_menu_attach_to_widget ()
void	gtk_menu_detach ()
GtkWidget *	gtk_menu_get_attach_widget ()
GList *	gtk_menu_get_for_attach_widget ()
void	(*GtkMenuPositionFunc) ()
void	(*GtkMenuDetachFunc) ()

Properties

GtkAccelGroup *	accel-group	Read / Write
gchar *	accel-path	Read / Write
gint	active	Read / Write
GdkAnchorHints	anchor-hints	Read / Write / Construct
GtkWidget *	attach-widget	Read / Write
GdkWindowTypeHint	menu-type-hint	Read / Write / Construct
gint	monitor	Read / Write
gint	rect-anchor-dx	Read / Write / Construct
gint	rect-anchor-dy	Read / Write / Construct
gboolean	reserve-toggle-size	Read / Write
gboolean	tearoff-state	Read / Write
gchar *	tearoff-title	Read / Write

Child Properties

gint	bottom-attach	Read / Write
gint	left-attach	Read / Write
gint	right-attach	Read / Write
gint	top-attach	Read / Write

Style Properties

GtkArrowPlacement	arrow-placement	Read
gfloat	arrow-scaling	Read
gboolean	double-arrows	Read
gint	horizontal-offset	Read
gint	horizontal-padding	Read
gint	vertical-offset	Read
gint	vertical-padding	Read

Signals

void	move-scroll	Action
void	popped-up	Run First

Types and Values

struct	GtkMenu
enum	GtkArrowPlacement

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkContainer
            └── GtkMenuShell
                └── GtkMenu
                    └── GtkRecentChooserMenu
```

Implemented Interfaces

GtkMenu implements AtkImplementorIface and GtkBuildable.

Includes

```
#include <gtk/gtk.h>
```

Description

A GtkMenu is a GtkMenuShell that implements a drop down menu consisting of a list of GtkMenuItem objects which can be navigated and activated by the user to perform application functions.

A GtkMenu is most commonly dropped down by activating a GtkMenuItem in a GtkMenuBar or popped up by activating a GtkMenuItem in another GtkMenu.

A GtkMenu can also be popped up by activating a GtkComboBox. Other composite widgets such as the GtkNotebook can pop up a GtkMenu as well.

Applications can display a GtkMenu as a popup menu by calling the `gtk_menu_popup()` function. The example below shows how an application can pop up a menu when the 3rd mouse button is pressed.

Connecting the popup signal handler.

```
1 // connect our handler which will popup the
2 menu
3 g_signal_connect_swapped (window,
  "button_press_event",
  G_CALLBACK (my_popup_handler), menu);
```

Signal handler which displays a popup menu.

```
1 static gint
2 my_popup_handler (GtkWidget *widget, GdkEvent
3 *event)
4 {
5   GtkMenu *menu;
6   GdkEventButton *event_button;
```

```
  g_return_val_if_fail (widget != NULL,
FALSE);
  g_return_val_if_fail (GTK_IS_MENU (widget),
FALSE);
  g_return_val_if_fail (event != NULL,
FALSE);

  // The "widget" is the menu that was
supplied when
  // g_signal_connect_swapped() was called.
  menu = GTK_MENU (widget);

  if (event->type == GDK_BUTTON_PRESS)
    {
      event_button = (GdkEventButton *)
event;
      if (event_button->button ==
GDK_BUTTON_SECONDARY)
        {
          gtk_menu_popup (menu, NULL, NULL,
NULL, NULL,
                          event_button-
>button, event_button->time);
          return TRUE;
        }
    }

  return FALSE;
}
```

CSS nodes

```
menu
├── arrow.top
├── <child>
┊
├── <child>
└── arrow.bottom
```

The main CSS node of GtkMenu has name menu, and there are two subnodes with name arrow, for scrolling menu arrows. These subnodes get the .top and .bottom style classes.

Functions

gtk_menu_new ()

```
GtkWidget *
gtk_menu_new (void);
```

Creates a new GtkMenu

Returns

a new GtkMenu

---

gtk_menu_new_from_model ()

```
GtkWidget *
gtk_menu_new_from_model (GMenuModel *model);
```

Creates a GtkMenu and populates it with menu items and submenus according to `model`.

The created menu items are connected to actions found in the GtkApplicationWindow to which the menu belongs - typically by means of being attached to a widget (see `gtk_menu_attach_to_widget()`) that is contained within the GtkApplicationWindows widget hierarchy.

Actions can also be added using `gtk_widget_insert_action_group()` on the menu's attach widget or on any of its parent widgets.

Parameters

model	a GMenuModel	

Returns

a new GtkMenu

Since: 3.4

---

gtk_menu_set_screen ()

```
void
gtk_menu_set_screen (GtkMenu *menu,
                     GdkScreen *screen);
```

Sets the GdkScreen on which the menu will be displayed.

Parameters

menu	a GtkMenu	
screen	a GdkScreen, or `NULL` if the screen should be determined by the widget the menu is attached to.	[allow-none]

Since: 2.2

---

gtk_menu_reorder_child ()

```
void
gtk_menu_reorder_child (GtkMenu *menu,
                        GtkWidget *child,
                        gint position);
```

Moves `child` to a new `position` in the list of `menu` children.

Parameters

menu	a GtkMenu
child	the GtkMenuItem to move
position	the new position to place `child` . Positions are numbered from 0 to n - 1

---

gtk_menu_attach ()

```
void
gtk_menu_attach (GtkMenu *menu,
                 GtkWidget *child,
                 guint left_attach,
                 guint right_attach,
                 guint top_attach,
                 guint bottom_attach);
```

Adds a new GtkMenuItem to a (table) menu. The number of “cells” that an item will occupy is specified by `left_attach` , `right_attach` , `top_attach` and `bottom_attach` . These each represent the leftmost, rightmost, uppermost and lower column and row numbers of the table. (Columns and rows are indexed from zero).

Note that this function is not related to `gtk_menu_detach()`.

Parameters

menu	a GtkMenu
child	a GtkMenuItem
left_attach	The column number to attach the left side of the item to
right_attach	The column number to attach the right side of the item to
top_attach	The row number to attach the top of the item to
bottom_attach	The row number to attach the bottom of the item to

Since: 2.4

---

gtk_menu_popup_at_rect ()

```
void
gtk_menu_popup_at_rect (GtkMenu *menu,
                        GdkWindow *rect_window,
                        const GdkRectangle *rect,
                        GdkGravity rect_anchor,
                        GdkGravity menu_anchor,
                        const GdkEvent *trigger_event);
```

Displays `menu` and makes it available for selection.

See `gtk_menu_popup_at_widget()` and `gtk_menu_popup_at_pointer()`, which handle more common cases for popping up menus.

`menu` will be positioned at `rect` , aligning their anchor points. `rect` is relative to the top-left corner of `rect_window` . `rect_anchor` and `menu_anchor` determine anchor points on `rect` and `menu` to pin together. `menu` can optionally be offset by “rect-anchor-dx” and “rect-anchor-dy”.

Anchors should be specified under the assumption that the text direction is left-to-right; they will be flipped horizontally automatically if the text direction is right-to-left.

Other properties that influence the behaviour of this function are “anchor-hints” and “menu-type-hint”. Connect to the “popped-up” signal to find out how it was actually positioned.

Parameters

menu	the GtkMenu to pop up	
rect_window	the GdkWindow `rect` is relative to.	[not nullable]
rect	the GdkRectangle to align `menu` with.	[not nullable]
rect_anchor	the point on `rect` to align with `menu` 's anchor point	
menu_anchor	the point on `menu` to align with `rect` 's anchor point	
trigger_event	the GdkEvent that initiated this request or `NULL` if it's the current event.	[nullable]

Since: 3.22

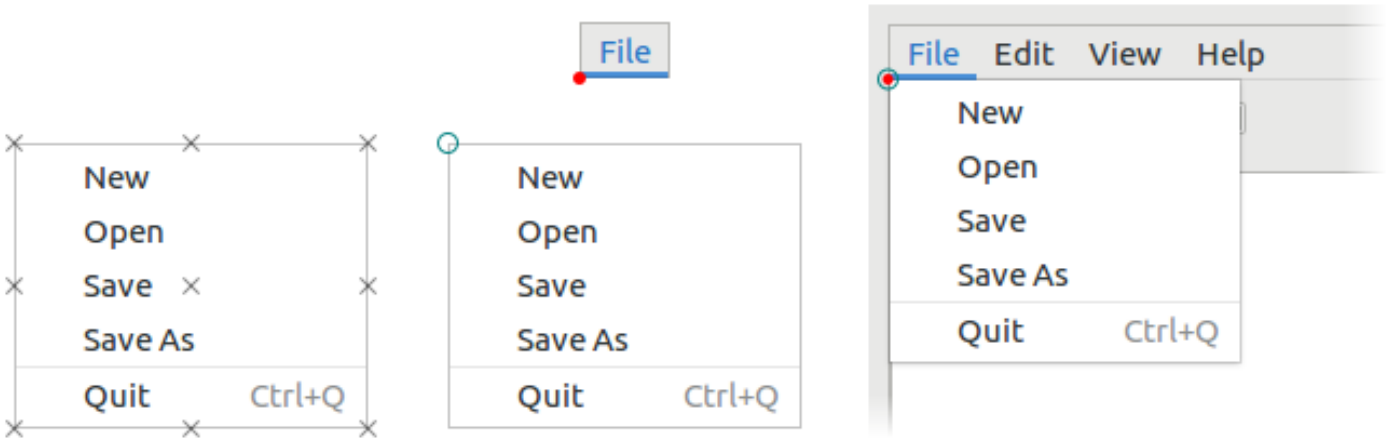
---

gtk_menu_popup_at_widget ()

```
void
gtk_menu_popup_at_widget (GtkMenu *menu,
                          GtkWidget *widget,
                          GdkGravity widget_anchor,
                          GdkGravity menu_anchor,
                          const GdkEvent *trigger_event);
```

Displays `menu` and makes it available for selection.

See `gtk_menu_popup_at_pointer()` to pop up a menu at the master pointer. `gtk_menu_popup_at_rect()` also allows you to position a menu at an arbitrary rectangle.



menu will be positioned at widget , aligning their anchor points. widget_anchor and menu_anchor determine anchor points on widget and menu to pin together. menu can optionally be offset by “rect-anchor-dx” and “rect-anchor-dy”.

Anchors should be specified under the assumption that the text direction is left-to-right; they will be flipped horizontally automatically if the text direction is right-to-left.

Other properties that influence the behaviour of this function are “anchor-hints” and “menu-type-hint”. Connect to the “popped-up” signal to find out how it was actually positioned.

Parameters

menu	the GtkMenu to pop up	
widget	the GtkWidget to align menu with.	[not nullable]
widget_anchor	the point on widget to align with menu 's anchor point	
menu_anchor	the point on menu to align with widget 's anchor point	
trigger_event	the GdkEvent that initiated this request or NULL if it's the current event.	[nullable]

Since: 3.22

---

gtk_menu_popup_at_pointer ()

```
void
gtk_menu_popup_at_pointer (GtkMenu *menu,
                           const GdkEvent *trigger_event);
```

Displays menu and makes it available for selection.

See gtk_menu_popup_at_widget() to pop up a menu at a widget. gtk_menu_popup_at_rect() also allows you to position a menu at an arbitrary rectangle.

menu will be positioned at the pointer associated with trigger_event .

Properties that influence the behaviour of this function are “anchor-hints”, “rect-anchor-dx”, “rect-anchor-dy”, and “menu-type-hint”. Connect to the “popped-up” signal to find out how it was actually positioned.

Parameters

menu	the GtkMenu to pop up	
trigger_event	the GdkEvent that initiated this request or NULL if it's the current event.	[nullable]

Since: 3.22

---

gtk_menu_popup_for_device ()

```
void
gtk_menu_popup_for_device (GtkMenu *menu,
                           GdkDevice *device,
                           GtkWidget *parent_menu_shell,
                           GtkWidget *parent_menu_item,
                           GtkMenuPositionFunc func,
                           gpointer data,
                           GDestroyNotify destroy,
                           guint button,
                           guint32 activate_time);
```

`gtk_menu_popup_for_device` has been deprecated since version 3.22 and should not be used in newly-written code.

Please use `gtk_menu_popup_at_widget()`, `gtk_menu_popup_at_pointer()`. or `gtk_menu_popup_at_rect()` instead

Displays a menu and makes it available for selection.

Applications can use this function to display context-sensitive menus, and will typically supply `NULL` for the `parent_menu_shell` , `parent_menu_item` , `func` , `data` and `destroy` parameters. The default menu positioning function will position the menu at the current position of `device` (or its corresponding pointer).

The `button` parameter should be the mouse button pressed to initiate the menu popup. If the menu popup was initiated by something other than a mouse button press, such as a mouse button release or a keypress, `button` should be 0.

The `activate_time` parameter is used to conflict-resolve initiation of concurrent requests for mouse/keyboard grab requests. To function properly, this needs to be the time stamp of the user event (such as a mouse click or key press) that caused the initiation of the popup. Only if no such event is available, `gtk_get_current_event_time()` can be used instead.

Note that this function does not work very well on GDK backends that do not have global coordinates, such as Wayland or Mir. You should probably use one of the gtk_menu_popup_at_ variants, which do not have this problem.

Parameters

menu	a GtkMenu	
device	a GdkDevice.	[allow-none]
parent_menu_shell	the menu shell containing the triggering menu item, or `NULL`.	[allow-none]
parent_menu_item	the menu item whose activation triggered the popup, or `NULL`.	[allow-none]
func	a user supplied function used to position the menu, or `NULL`.	[allow-none]
data	user supplied data to be passed to `func` .	[allow-none]
destroy	destroy notify for `data` .	[allow-none]
button	the mouse button which was pressed to initiate the event	
activate_time	the time at which the activation event occurred	

Since: 3.0

gtk_menu_popup ()

```
void
gtk_menu_popup (GtkMenu *menu,
                GtkWidget *parent_menu_shell,
                GtkWidget *parent_menu_item,
                GtkMenuPositionFunc func,
                gpointer data,
                guint button,
                guint32 activate_time);
```

`gtk_menu_popup` has been deprecated since version 3.22 and should not be used in newly-written code.

Please use `gtk_menu_popup_at_widget()`, `gtk_menu_popup_at_pointer()`. or `gtk_menu_popup_at_rect()` instead

Displays a menu and makes it available for selection.

Applications can use this function to display context-sensitive menus, and will typically supply `NULL` for the `parent_menu_shell` , `parent_menu_item` , `func` and `data` parameters. The default menu positioning function will position the menu at the current mouse cursor position.

The `button` parameter should be the mouse button pressed to initiate the menu popup. If the menu popup was initiated by something other than a mouse button press, such as a mouse button release or a keypress, `button` should be 0.

The `activate_time` parameter is used to conflict-resolve initiation of concurrent requests for mouse/keyboard grab requests. To function properly, this needs to be the timestamp of the user event (such as a mouse click or key press) that caused the initiation of the popup. Only if no such event is available, `gtk_get_current_event_time()` can be used instead.

Note that this function does not work very well on GDK backends that do not have global coordinates, such as Wayland or Mir. You should probably use one of the gtk_menu_popup_at_ variants, which do not have this problem.

Parameters

menu	a GtkMenu	
parent_menu_shell	the menu shell containing the triggering menu item, or `NULL`.	[allow-none]
parent_menu_item	the menu item whose activation triggered the popup, or `NULL`.	[allow-none]
func	a user supplied function used to position the menu, or `NULL`.	[scope async][allow-none]
data	user supplied data to be passed to `func` .	
button	the mouse button which was pressed to initiate the event.	
activate_time	the time at which the activation event occurred.	

gtk_menu_set_accel_group ()

```
void
gtk_menu_set_accel_group (GtkMenu *menu,
                          GtkAccelGroup *accel_group);
```

Set the GtkAccelGroup which holds global accelerators for the menu. This accelerator group needs to also be added to all windows that this menu is being used in with `gtk_window_add_accel_group()`, in order for those windows to support all the accelerators contained in this group.

Parameters

menu	a GtkMenu
accel_group	the GtkAccelGroup to be associated with the menu. [allow-none]

---

gtk_menu_get_accel_group ()

```
GtkAccelGroup *
gtk_menu_get_accel_group (GtkMenu *menu);
```

Gets the GtkAccelGroup which holds global accelerators for the menu. See `gtk_menu_set_accel_group()`.

Parameters

menu	a GtkMenu

Returns

the GtkAccelGroup associated with the menu.

[transfer none]

---

gtk_menu_set_accel_path ()

```
void
gtk_menu_set_accel_path (GtkMenu *menu,
                         const gchar *accel_path);
```

Sets an accelerator path for this menu from which accelerator paths for its immediate children, its menu items, can be constructed. The main purpose of this function is to spare the programmer the inconvenience of having to call `gtk_menu_item_set_accel_path()` on each menu item that should support runtime user changable accelerators. Instead, by just calling `gtk_menu_set_accel_path()` on their parent, each menu item of this menu, that contains a label describing its purpose, automatically gets an accel path assigned.

For example, a menu containing menu items “New” and “Exit”, will, after `gtk_menu_set_accel_path (menu, "&lt;Gnumeric-Sheet>/File");` has been called, assign its items the accel paths: `"&lt;Gnumeric-Sheet>/File/New"` and `"&lt;Gnumeric-Sheet>/File/Exit"`.

Assigning accel paths to menu items then enables the user to change their accelerators at runtime. More details

about accelerator paths and their default setups can be found at `gtk_accel_map_add_entry()`.

Note that `accel_path` string will be stored in a GQuark. Therefore, if you pass a static string, you can save some memory by interning it first with `g_intern_static_string()`.

Parameters

menu	a valid GtkMenu	
accel_path	a valid accelerator path, or `NULL` to unset the path.	[nullable]

---

gtk_menu_get_accel_path ()

```
const gchar *
gtk_menu_get_accel_path (GtkMenu *menu);
```

Retrieves the accelerator path set on the menu.

Parameters

menu	a valid GtkMenu	

Returns

the accelerator path set on the menu.

Since: 2.14

---

gtk_menu_set_title ()

```
void
gtk_menu_set_title (GtkMenu *menu,
                    const gchar *title);
```

`gtk_menu_set_title` has been deprecated since version 3.10 and should not be used in newly-written code.

Sets the title string for the menu.

The title is displayed when the menu is shown as a tearoff menu. If `title` is `NULL`, the menu will see if it is attached to a parent menu item, and if so it will try to use the same text as that menu item's label.

Parameters

menu	a GtkMenu	
title	a string containing the title for the menu, or `NULL` to inherit the title of the parent menu item, if any.	[nullable]

---

gtk_menu_get_title ()

```
const gchar *
gtk_menu_get_title (GtkMenu *menu);
```

`gtk_menu_get_title` has been deprecated since version 3.10 and should not be used in newly-written code.

Returns the title of the menu. See `gtk_menu_set_title()`.

Parameters

menu	a GtkMenu

Returns

the title of the menu, or `NULL` if the menu has no title set on it. This string is owned by GTK+ and should not be modified or freed.

---

gtk_menu_set_monitor ()

```
void
gtk_menu_set_monitor (GtkMenu *menu,
                      gint monitor_num);
```

Informs GTK+ on which monitor a menu should be popped up. See `gdk_monitor_get_geometry()`.

This function should be called from a GtkMenuPositionFunc if the menu should not appear on the same monitor as the pointer. This information can't be reliably inferred from the coordinates returned by a GtkMenuPositionFunc, since, for very long menus, these coordinates may extend beyond the monitor boundaries or even the screen boundaries.

Parameters

menu	a GtkMenu
monitor_num	the number of the monitor on which the menu should be popped up

Since: 2.4

---

gtk_menu_get_monitor ()

```
gint
gtk_menu_get_monitor (GtkMenu *menu);
```

Retrieves the number of the monitor on which to show the menu.

Parameters

menu	a GtkMenu

Returns

the number of the monitor on which the menu should be popped up or -1, if no monitor has been set

Since: 2.14

---

gtk_menu_place_on_monitor ()

```
void
gtk_menu_place_on_monitor (GtkMenu *menu,
                           GdkMonitor *monitor);
```

Places `menu` on the given monitor.

Parameters

menu	a GtkMenu
monitor	the monitor to place the menu on

Since: 3.22

---

gtk_menu_get_tearoff_state ()

```
gboolean
gtk_menu_get_tearoff_state (GtkMenu *menu);
```

`gtk_menu_get_tearoff_state` has been deprecated since version 3.10 and should not be used in newly-written code.

Returns whether the menu is torn off. See `gtk_menu_set_tearoff_state()`.

Parameters

menu	a GtkMenu

Returns

`TRUE` if the menu is currently torn off.

---

gtk_menu_set_reserve_toggle_size ()

```
void
gtk_menu_set_reserve_toggle_size (GtkMenu *menu,
                                  gboolean reserve_toggle_size);
```

Sets whether the menu should reserve space for drawing toggles or icons, regardless of their actual presence.

Parameters

menu	a GtkMenu
reserve_toggle_size	whether to reserve size for toggles

Since: 2.18

---

gtk_menu_get_reserve_toggle_size ()

```
gboolean
gtk_menu_get_reserve_toggle_size (GtkMenu *menu);
```

Returns whether the menu reserves space for toggles and icons, regardless of their actual presence.

Parameters

menu	a GtkMenu

Returns

Whether the menu reserves toggle space

Since: 2.18

---

gtk_menu_popdown ()

```
void
gtk_menu_popdown (GtkMenu *menu);
```

Removes the menu from the screen.

Parameters

menu	a GtkMenu

---

gtk_menu_reposition ()

```
void
gtk_menu_reposition (GtkMenu *menu);
```

Repositions the menu according to its position function.

Parameters

menu	a GtkMenu

---

gtk_menu_get_active ()

```
GtkWidget *
gtk_menu_get_active (GtkMenu *menu);
```

Returns the selected menu item from the menu. This is used by the GtkComboBox.

Parameters

menu	a GtkMenu

Returns

the GtkMenuItem that was last selected in the menu. If a selection has not yet been made, the first menu item is selected.

[transfer none]

---

gtk_menu_set_active ()

```
void
gtk_menu_set_active (GtkMenu *menu,
                     guint index);
```

Selects the specified menu item within the menu. This is used by the GtkComboBox and should not be used by anyone else.

Parameters

menu	a GtkMenu
index	the index of the menu item to select. Index values are from 0 to n-1

---

gtk_menu_set_tearoff_state ()

```
void
gtk_menu_set_tearoff_state (GtkMenu *menu,
                            gboolean torn_off);
```

`gtk_menu_set_tearoff_state` has been deprecated since version 3.10 and should not be used in newly-written code.

Changes the tearoff state of the menu. A menu is normally displayed as drop down menu which persists as long as the menu is active. It can also be displayed as a tearoff menu which persists until it is closed or reattached.

Parameters

menu	a GtkMenu
torn_off	If `TRUE`, menu is displayed as a tearoff menu.

gtk_menu_attach_to_widget ()

```
void
gtk_menu_attach_to_widget (GtkMenu *menu,
                           GtkWidget *attach_widget,
                           GtkMenuDetachFunc detacher);
```

Attaches the menu to the widget and provides a callback function that will be invoked when the menu calls `gtk_menu_detach()` during its destruction.

If the menu is attached to the widget then it will be destroyed when the widget is destroyed, as if it was a child widget. An attached menu will also move between screens correctly if the widgets moves between screens.

Parameters

menu	a GtkMenu	
attach_widget	the GtkWidget that the menu will be attached to	
detacher	the user supplied callback function that will be called when the menu calls `gtk_menu_detach()`.	[scope async][allow-none]

gtk_menu_detach ()

```
void
gtk_menu_detach (GtkMenu *menu);
```

Detaches the menu from the widget to which it had been attached. This function will call the callback function, `detacher` , provided when the `gtk_menu_attach_to_widget()` function was called.

Parameters

menu	a GtkMenu

gtk_menu_get_attach_widget ()

```
GtkWidget *
gtk_menu_get_attach_widget (GtkMenu *menu);
```

Returns the GtkWidget that the menu is attached to.

Parameters

menu	a GtkMenu

Returns

the GtkWidget that the menu is attached to.

[transfer none]

---

gtk_menu_get_for_attach_widget ()

```
GList *
gtk_menu_get_for_attach_widget (GtkWidget *widget);
```

Returns a list of the menus which are attached to this widget. This list is owned by GTK+ and must not be modified.

Parameters

widget	a GtkWidget	

Returns

the list of menus attached to his widget.

[element-type GtkWidget][transfer none]

Since: 2.6

GtkMenuPositionFunc ()

```
void
(*GtkMenuPositionFunc) (GtkMenu *menu,
                        gint *x,
                        gint *y,
                        gboolean *push_in,
                        gpointer user_data);
```

A user function supplied when calling `gtk_menu_popup()` which controls the positioning of the menu when it is displayed. The function sets the `x` and `y` parameters to the coordinates where the menu is to be drawn. To make the menu appear on a different monitor than the mouse pointer, `gtk_menu_set_monitor()` must be called.

Parameters

menu	a GtkMenu.	
x	address of the gint representing the horizontal position where the menu shall be drawn.	[inout]
y	address of the gint representing the vertical position where the menu shall be drawn. This is an output parameter.	[inout]
push_in	This parameter controls how menus	[out]

	placed outside the monitor are handled. If this is set to `TRUE` and part of the menu is outside the monitor then GTK+ pushes the window into the visible area, effectively modifying the popup position. Note that moving and possibly resizing the menu around will alter the scroll position to keep the menu items “in place”, i.e. at the same monitor position they would have been without resizing. In practice, this behavior is only useful for combobox popups or option menus and cannot be used to simply confine a menu to monitor boundaries. In that case, changing the scroll offset is not desirable.
user_data	the data supplied by the user in the `gtk_menu_popup()` `data` parameter.

GtkMenuDetachFunc ()

```
void
(*GtkMenuDetachFunc) (GtkWidget *attach_widget,
                      GtkMenu *menu);
```

A user function supplied when calling `gtk_menu_attach_to_widget()` which will be called when the menu is later detached from the widget.

Parameters

attach_widget	the GtkWidget that the menu is being detached from.
menu	the GtkMenu being detached.

Types and Values

struct GtkMenu

```
struct GtkMenu;
```

---

enum GtkArrowPlacement

Used to specify the placement of scroll arrows in scrolling menus.

Members

GTK_ARROWS_BOTH	Place one arrow on each end of the menu.
GTK_ARROWS_START	Place both arrows at the top of the menu.
GTK_ARROWS_END	Place both arrows at the bottom of the menu.

Property Details

The `"accel-group"` property

`"accel-group"` `GtkAccelGroup *`

The accel group holding accelerators for the menu.

Flags: Read / Write

Since: 2.14

The `"accel-path"` property

`"accel-path"` `gchar *`

An accel path used to conveniently construct accel paths of child items.

Flags: Read / Write

Default value: NULL

Since: 2.14

The `"active"` property

`"active"` `gint`

The index of the currently selected menu item, or -1 if no menu item is selected.

Flags: Read / Write

Allowed values: >= -1

Default value: -1

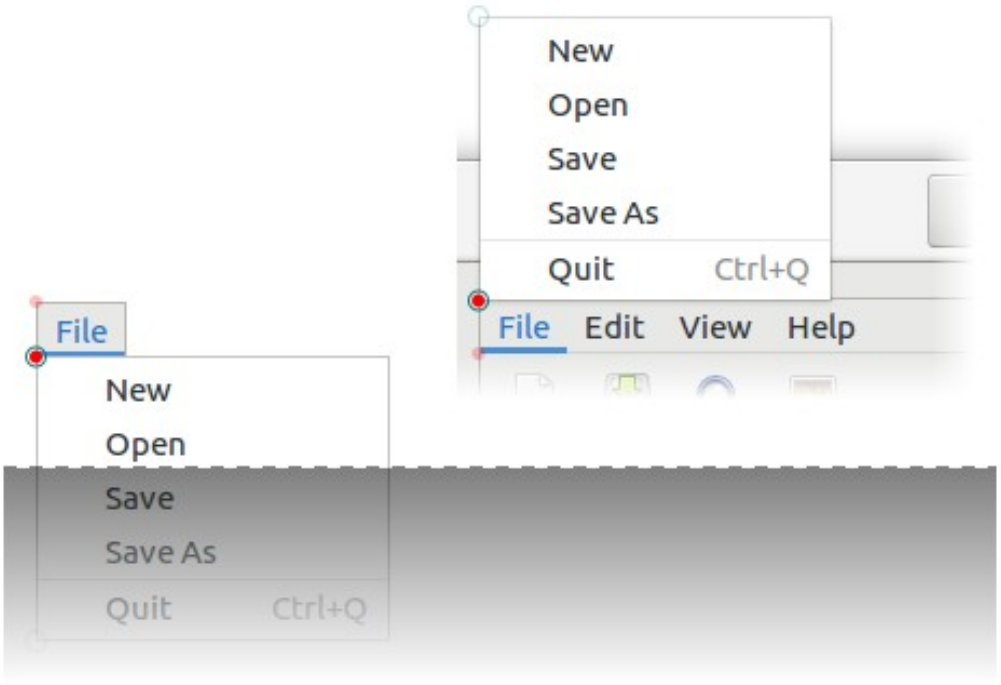
Since: 2.14

The “anchor-hints” property

“anchor-hints” GdkAnchorHints

Positioning hints for aligning the menu relative to a rectangle.

These hints determine how the menu should be positioned in the case that the menu would fall off-screen if placed in its ideal position.



For example, GDK_ANCHOR_FLIP_Y will replace GDK_GRAVITY_NORTH_WEST with GDK_GRAVITY_SOUTH_WEST and vice versa if the menu extends beyond the bottom edge of the monitor.

See gtk_menu_popup_at_rect(), gtk_menu_popup_at_widget(), gtk_menu_popup_at_pointer(), “rect-anchor-dx”, “rect-anchor-dy”, “menu-type-hint”, and “popped-up”.

Flags: Read / Write / Construct

Default value: GDK_ANCHOR_FLIP_X | GDK_ANCHOR_FLIP_Y | GDK_ANCHOR_SLIDE_X | GDK_ANCHOR_SLIDE_Y | GDK_ANCHOR_RESIZE_X | GDK_ANCHOR_RESIZE_Y

Since: 3.22

The “attach-widget” property

“attach-widget” GtkWidget *

The widget the menu is attached to. Setting this property attaches the menu without a GtkMenuDetachFunc. If you need to use a detacher, use gtk_menu_attach_to_widget() directly.

Flags: Read / Write

Since: 2.14

The “menu-type-hint” property

“menu-type-hint” GdkWindowTypeHint

The GdkWindowTypeHint to use for the menu's GdkWindow.

See gtk_menu_popup_at_rect(), gtk_menu_popup_at_widget(), gtk_menu_popup_at_pointer(), “anchor-hints”, “rect-anchor-dx”, “rect-anchor-dy”, and “popped-up”.

Flags: Read / Write / Construct

Default value: GDK_WINDOW_TYPE_HINT_POPUP_MENU

Since: 3.22

The “monitor” property

“monitor” gint

The monitor the menu will be popped up on.

Flags: Read / Write

Allowed values: >= -1

Default value: -1

Since: 2.14

The “rect-anchor-dx” property

“rect-anchor-dx” gint

Horizontal offset to apply to the menu, i.e. the rectangle or widget anchor.

See gtk_menu_popup_at_rect(), gtk_menu_popup_at_widget(), gtk_menu_popup_at_pointer(), “anchor-hints”, “rect-anchor-dy”, “menu-type-hint”, and “popped-up”.

Flags: Read / Write / Construct

Default value: 0

Since: 3.22

The “rect-anchor-dy” property

“rect-anchor-dy” gint

Vertical offset to apply to the menu, i.e. the rectangle or widget anchor.

See gtk_menu_popup_at_rect(), gtk_menu_popup_at_widget(), gtk_menu_popup_at_pointer(), “anchor-hints”, “rect-anchor-dx”, “menu-type-hint”, and “popped-up”.

Flags: Read / Write / Construct

Default value: 0

Since: 3.22

The “reserve-toggle-size” property

“reserve-toggle-size” gboolean

A boolean that indicates whether the menu reserves space for toggles and icons, regardless of their actual presence.

This property should only be changed from its default value for special-purposes such as tabular menus. Regular menus that are connected to a menu bar or context menus should reserve toggle space for consistency.

Flags: Read / Write

Default value: TRUE

Since: 2.18

The “tearoff-state” property

“tearoff-state” gboolean

A boolean that indicates whether the menu is torn-off.

GtkMenu:tearoff-state has been deprecated since version 3.10 and should not be used in newly-written code.

Flags: Read / Write

Default value: FALSE

Since: 2.6

The “tearoff-title” property

“tearoff-title” gchar *

A title that may be displayed by the window manager when this menu is torn-off.

GtkMenu:tearoff-title has been deprecated since version 3.10 and should not be used in newly-written code.

Flags: Read / Write

Default value: NULL

Child Property Details

The “bottom-attach” child property

“bottom-attach” gint

The row number to attach the bottom of the child to.

Flags: Read / Write

Allowed values: >= -1

Default value: -1

The “left-attach” child property

“left-attach” gint

The column number to attach the left side of the child to.

Flags: Read / Write

Allowed values: >= -1

Default value: -1

The “right-attach” child property

“right-attach” gint

The column number to attach the right side of the child to.

Flags: Read / Write

Allowed values: >= -1

Default value: -1

The “top-attach” child property

“top-attach” gint

The row number to attach the top of the child to.

Flags: Read / Write

Allowed values: >= -1

Default value: -1

Style Property Details

The “arrow-placement” style property

“arrow-placement” GtkArrowPlacement

Indicates where scroll arrows should be placed.

GtkMenu:arrow-placement has been deprecated since version 3.20 and should not be used in newly-written

code.

the value of this style property is ignored.

Flags: Read

Default value: GTK_ARROWS_BOTH

Since: 2.16

The “arrow-scaling” style property

“arrow-scaling” gfloat

Arbitrary constant to scale down the size of the scroll arrow.

GtkMenu:arrow-scaling has been deprecated since version 3.20 and should not be used in newly-written code.

use the standard min-width/min-height CSS properties on the arrow node; the value of this style property is ignored.

Flags: Read

Allowed values: [0,1]

Default value: 0.7

Since: 2.16

The “double-arrows” style property

“double-arrows” gboolean

When TRUE, both arrows are shown when scrolling.

GtkMenu:double-arrows has been deprecated since version 3.20 and should not be used in newly-written code.

the value of this style property is ignored.

Flags: Read

Default value: TRUE

The “horizontal-offset” style property

“horizontal-offset” gint

When the menu is a submenu, position it this number of pixels offset horizontally.

Flags: Read

Default value: -2

The “horizontal-padding” style property

“horizontal-padding” gint

Extra space at the left and right edges of the menu.

GtkMenu:horizontal-padding has been deprecated since version 3.8 and should not be used in newly-written code.

use the standard padding CSS property (through objects like GtkStyleContext and GtkCssProvider); the value of this style property is ignored.

Flags: Read

Allowed values: >= 0

Default value: 0

The “vertical-offset” style property

“vertical-offset” gint

When the menu is a submenu, position it this number of pixels offset vertically.

Flags: Read

Default value: 0

The “vertical-padding” style property

“vertical-padding” gint

Extra space at the top and bottom of the menu.

GtkMenu:vertical-padding has been deprecated since version 3.8 and should not be used in newly-written code.

use the standard padding CSS property (through objects like GtkStyleContext and GtkCssProvider); the value of this style property is ignored.

Flags: Read

Allowed values: >= 0

Default value: 1

Signal Details

The “move-scroll” signal

```
void
user_function (GtkMenu      *menu,
               GtkScrollType scroll_type,
               gpointer      user_data)
```

Parameters

menu	a GtkMenu
scroll_type	a GtkScrollType
user_data	user data set when the signal handler was connected.

Flags: Action

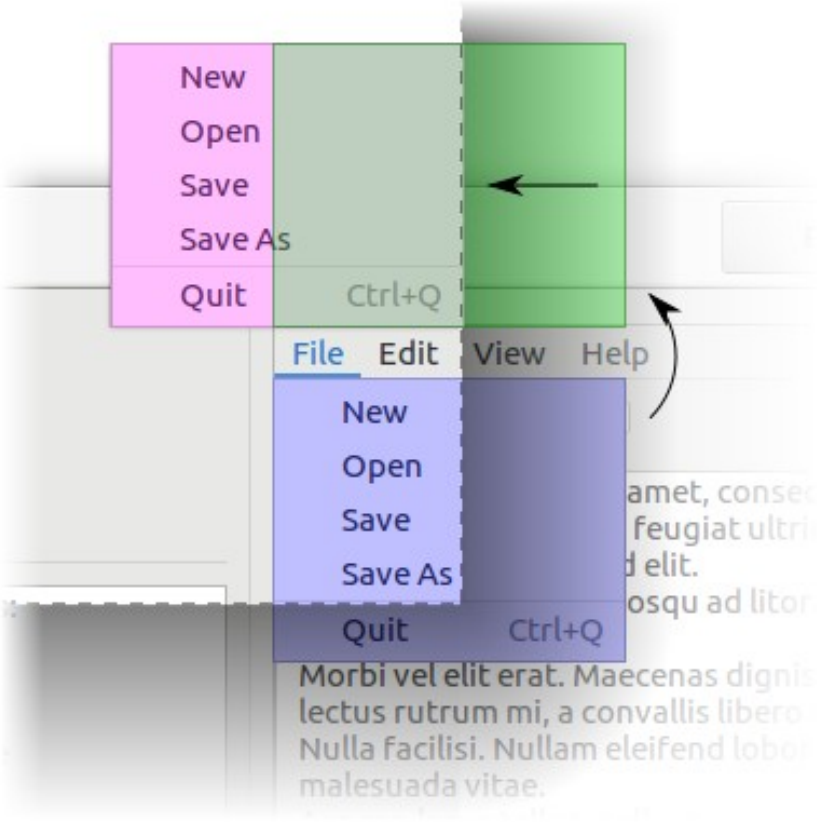
The “popped-up” signal

```
void
user_function (GtkMenu *menu,
               gpointer flipped_rect,
               gpointer final_rect,
               gboolean flipped_x,
               gboolean flipped_y,
               gpointer user_data)
```

Emitted when the position of `menu` is finalized after being popped up using `gtk_menu_popup_at_rect()`, `gtk_menu_popup_at_widget()`, or `gtk_menu_popup_at_pointer()`.

`menu` might be flipped over the anchor rectangle in order to keep it on-screen, in which case `flipped_x` and `flipped_y` will be set to `TRUE` accordingly.

`flipped_rect` is the ideal position of `menu` after any possible flipping, but before any possible sliding. `final_rect` is `flipped_rect` , but possibly translated in the case that flipping is still ineffective in keeping `menu` on-screen.



The blue menu is `menu` 's ideal position, the green menu is `flipped_rect` , and the red menu is `final_rect` .

See `gtk_menu_popup_at_rect()`, `gtk_menu_popup_at_widget()`, `gtk_menu_popup_at_pointer()`, “anchor-hints”, “rect-anchor-dx”, “rect-anchor-dy”, and “menu-type-hint”.

Parameters

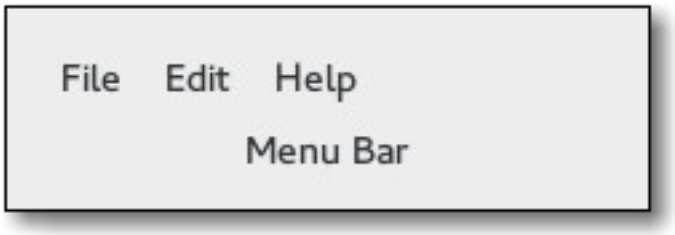
menu	the GtkMenu that popped up	
flipped_rect	the position of `menu` after any possible flipping or `NULL` if the backend can't obtain it.	[nullable]
final_rect	the final position of `menu` or `NULL` if the backend can't obtain it.	[nullable]
flipped_x	`TRUE` if the anchors were flipped horizontally	
flipped_y	`TRUE` if the anchors were flipped vertically	
user_data	user data set when the signal handler was connected.	

Flags: Run First

Since: 3.22

GtkMenuBar

GtkMenuBar — A subclass of GtkMenuShell which holds GtkMenuItem widgets



Functions

GtkWidget *	gtk_menu_bar_new ()
GtkWidget *	gtk_menu_bar_new_from_model ()
void	gtk_menu_bar_set_pack_direction ()
GtkPackDirection	gtk_menu_bar_get_pack_direction ()
void	gtk_menu_bar_set_child_pack_direction ()
GtkPackDirection	gtk_menu_bar_get_child_pack_direction ()

Properties

GtkPackDirection	child-pack-direction	Read / Write
GtkPackDirection	pack-direction	Read / Write

Style Properties

gint	internal-padding	Read

GtkShadowType	shadow-type	Read

Types and Values

struct	GtkMenuBar
enum	GtkPackDirection

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkContainer
            └── GtkMenuShell
                └── GtkMenuBar
```

Implemented Interfaces

GtkMenuBar implements AtkImplementorIface and GtkBuildable.

Includes

```
#include <gtk/gtk.h>
```

Description

The GtkMenuBar is a subclass of GtkMenuShell which contains one or more GtkMenuItems. The result is a standard menu bar which can hold many menu items.

CSS nodes

GtkMenuBar has a single CSS node with name menubar.

Functions

gtk_menu_bar_new ()

```
GtkWidget *
gtk_menu_bar_new (void);
```

Creates a new GtkMenuBar

Returns

the new menu bar, as a GtkWidget

gtk_menu_bar_new_from_model ()

```
GtkWidget *
gtk_menu_bar_new_from_model (GMenuModel *model);
```

Creates a new GtkMenuBar and populates it with menu items and submenus according to `model`.

The created menu items are connected to actions found in the GtkApplicationWindow to which the menu bar belongs - typically by means of being contained within the GtkApplicationWindows widget hierarchy.

Parameters

model	a GMenuModel

Returns

a new GtkMenuBar

Since: 3.4

---

gtk_menu_bar_set_pack_direction ()

```
void
gtk_menu_bar_set_pack_direction (GtkMenuBar *menubar,
                                 GtkPackDirection pack_dir);
```

Sets how items should be packed inside a menubar.

Parameters

menubar	a GtkMenuBar
pack_dir	a new GtkPackDirection

Since: 2.8

---

gtk_menu_bar_get_pack_direction ()

```
GtkPackDirection
gtk_menu_bar_get_pack_direction (GtkMenuBar *menubar);
```

Retrieves the current pack direction of the menubar. See `gtk_menu_bar_set_pack_direction()`.

Parameters

menubar	a GtkMenuBar

Returns

the pack direction

Since: 2.8

---

gtk_menu_bar_set_child_pack_direction ()

```
void
gtk_menu_bar_set_child_pack_direction (GtkMenuBar *menubar,
                                       GtkPackDirection child_pack_dir);
```

Sets how widgets should be packed inside the children of a menubar.

Parameters

menubar	a GtkMenuBar
child_pack_dir	a new GtkPackDirection

Since: 2.8

---

gtk_menu_bar_get_child_pack_direction ()

```
GtkPackDirection
gtk_menu_bar_get_child_pack_direction (GtkMenuBar *menubar);
```

Retrieves the current child pack direction of the menubar. See `gtk_menu_bar_set_child_pack_direction()`.

Parameters

menubar	a GtkMenuBar

Returns

the child pack direction

Since: 2.8

Types and Values

struct GtkMenuBar

```
struct GtkMenuBar;
```

---

enum GtkPackDirection

Determines how widgets should be packed inside menubars and menuitems contained in menubars.

Members

GTK_PACK_DIRECTION_LTR	Widgets are packed left-to-right
GTK_PACK_DIRECTION_RTL	Widgets are packed right-to-left
GTK_PACK_DIRECTION_TTB	Widgets are packed top-to-bottom
GTK_PACK_DIRECTION_BTT	Widgets are packed bottom-to-top

Property Details

The `“child-pack-direction”` property

`“child-pack-direction” GtkPackDirection`

The child pack direction of the menubar. It determines how the widgets contained in child menuitems are arranged.

Flags: Read / Write

Default value: GTK_PACK_DIRECTION_LTR

Since: 2.8

The `“pack-direction”` property

`“pack-direction” GtkPackDirection`

The pack direction of the menubar. It determines how menuitems are arranged in the menubar.

Flags: Read / Write

Default value: GTK_PACK_DIRECTION_LTR

Since: 2.8

Style Property Details

The `“internal-padding”` style property

`“internal-padding” gint`

Amount of border space between the menubar shadow and the menu items

`GtkMenuBar:internal-padding` has been deprecated since version 3.8 and should not be used in newly-written code.

use the standard padding CSS property (through objects like GtkStyleContext and GtkCssProvider); the value of this style property is ignored.

Flags: Read

Allowed values: >= 0

Default value: 0

The “shadow-type” style property

“shadow-type” GtkShadowType

The style of the shadow around the menubar.

GtkMenuBar:shadow-type has been deprecated since version 3.20 and should not be used in newly-written code.

Use CSS to determine the shadow; the value of this style property is ignored.

Flags: Read

Default value: GTK_SHADOW_OUT

See Also

GtkMenuShell, GtkMenu, GtkMenuItem

GtkMenuItem

GtkMenuItem — The widget used for item in menus

Functions

GtkWidget *	gtk_menu_item_new ()
GtkWidget *	gtk_menu_item_new_with_label ()
GtkWidget *	gtk_menu_item_new_with_mnemonic ()
void	gtk_menu_item_set_right_justified ()
gboolean	gtk_menu_item_get_right_justified ()
const gchar *	gtk_menu_item_get_label ()
void	gtk_menu_item_set_label ()
gboolean	gtk_menu_item_get_use_underline ()
void	gtk_menu_item_set_use_underline ()
void	gtk_menu_item_set_submenu ()
GtkWidget *	gtk_menu_item_get_submenu ()
void	gtk_menu_item_set_accel_path ()
const gchar *	gtk_menu_item_get_accel_path ()
void	gtk_menu_item_select ()
void	gtk_menu_item_deselect ()
void	gtk_menu_item_activate ()
void	gtk_menu_item_toggle_size_request ()
void	gtk_menu_item_toggle_size_allocate ()
gboolean	gtk_menu_item_get_reserve_indicator ()
void	gtk_menu_item_set_reserve_indicator ()

Properties

gchar *	accel-path	Read / Write

gchar *	label	Read / Write
gboolean	right-justified	Read / Write
GtkMenu *	submenu	Read / Write
gboolean	use-underline	Read / Write

Style Properties

gfloat	arrow-scaling	Read
gint	arrow-spacing	Read
gint	horizontal-padding	Read
GtkShadowType	selected-shadow-type	Read
gint	toggle-spacing	Read
gint	width-chars	Read

Signals

void	activate	Action
void	activate-item	Run First
void	deselect	Run First
void	select	Run First
void	toggle-size-allocate	Run First
void	toggle-size-request	Run First

Types and Values

struct	GtkMenuItem
struct	GtkMenuItemClass

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkContainer
            └── GtkBin
                └── GtkMenuItem
                    ├── GtkCheckMenuItem
                    ├── GtkImageMenuItem
                    ├── GtkSeparatorMenuItem
                    └── GtkTearoffMenuItem
```

Implemented Interfaces

GtkMenuItem implements AtkImplementorIface, GtkBuildable, GtkActivatable and GtkActionable.

Includes

```
#include <gtk/gtk.h>
```

Description

The GtkMenuItem widget and the derived widgets are the only valid children for menus. Their function is to correctly handle highlighting, alignment, events and submenus.

As a GtkMenuItem derives from GtkBin it can hold any valid child widget, although only a few are really useful.

By default, a GtkMenuItem sets a GtkAccelLabel as its child. GtkMenuItem has direct functions to set the label and its mnemonic. For more advanced label settings, you can fetch the child widget from the GtkBin.

An example for setting markup and accelerator on a MenuItem:

```
GtkWidget *menu_item =
gtk_menu_item_new_with_label ("Example Menu
Item");

GtkWidget *child = gtk_bin_get_child (GTK_BIN
(menu_item));
gtk_label_set_markup (GTK_LABEL (child),
"<i>new label</i> with <b>markup</b>");
gtk_accel_label_set_accel (GTK_ACCEL_LABEL
(child), GDK_KEY_1, 0);
```

GtkMenuItem as GtkBuildable

The GtkMenuItem implementation of the GtkBuildable interface supports adding a submenu by specifying “submenu” as the “type” attribute of a <child> element.

An example of UI definition fragment with submenus:

```
<object class="GtkMenuItem">
  <child type="submenu">
    <object class="GtkMenu"/>
  </child>
</object>
```

CSS nodes

```
menuitem
├── <child>
╰── [arrow.right]
```

GtkMenuItem has a single CSS node with name menuitem. If the menuitem has a submenu, it gets another CSS node with name arrow, which has the .left or .right style class.

Functions

gtk_menu_item_new ()

```
GtkWidget *
gtk_menu_item_new (void);
```

Creates a new GtkMenuItem.

Returns

the newly created GtkMenuItem

---

gtk_menu_item_new_with_label ()

```
GtkWidget *
gtk_menu_item_new_with_label (const gchar *label);
```

Creates a new GtkMenuItem whose child is a GtkLabel.

Parameters

label	the text for the label

Returns

the newly created GtkMenuItem

---

gtk_menu_item_new_with_mnemonic ()

```
GtkWidget *
gtk_menu_item_new_with_mnemonic (const gchar *label);
```

Creates a new GtkMenuItem containing a label.

The label will be created using `gtk_label_new_with_mnemonic()`, so underscores in `label` indicate the mnemonic for the menu item.

Parameters

label	The text of the button, with an underscore in front of the mnemonic character

Returns

a new GtkMenuItem

---

gtk_menu_item_set_right_justified ()

```
void
gtk_menu_item_set_right_justified (GtkMenuItem *menu_item,
                                   gboolean right_justified);
```

`gtk_menu_item_set_right_justified` has been deprecated since version 3.2 and should not be used in newly-written code.

If you insist on using it, use `gtk_widget_set_hexpand()` and `gtk_widget_set_halign()`.

Sets whether the menu item appears justified at the right side of a menu bar. This was traditionally done for “Help” menu items, but is now considered a bad idea. (If the widget layout is reversed for a right-to-left language like Hebrew or Arabic, right-justified-menu-items appear at the left.)

Parameters

menu_item	a GtkMenuItem.
right_justified	if `TRUE` the menu item will appear at the far right if added to a menu bar

---

gtk_menu_item_get_right_justified ()

```
gboolean
gtk_menu_item_get_right_justified (GtkMenuItem *menu_item);
```

`gtk_menu_item_get_right_justified` has been deprecated since version 3.2 and should not be used in newly-written code.

See `gtk_menu_item_set_right_justified()`

Gets whether the menu item appears justified at the right side of the menu bar.

Parameters

menu_item	a GtkMenuItem

Returns

`TRUE` if the menu item will appear at the far right if added to a menu bar.

---

gtk_menu_item_get_label ()

```
const gchar *
gtk_menu_item_get_label (GtkMenuItem *menu_item);
```

Sets `text` on the `menu_item` label

Parameters

menu_item	a GtkMenuItem

Returns

The text in the `menu_item` label. This is the internal string used by the label, and must not be modified.

Since: 2.16

---

gtk_menu_item_set_label ()

```
void
gtk_menu_item_set_label (GtkMenuItem *menu_item,
                         const gchar *label);
```

Sets `text` on the `menu_item` label

Parameters

menu_item	a GtkMenuItem
label	the text you want to set

Since: 2.16

---

gtk_menu_item_get_use_underline ()

```
gboolean
gtk_menu_item_get_use_underline (GtkMenuItem *menu_item);
```

Checks if an underline in the text indicates the next character should be used for the mnemonic accelerator key.

Parameters

menu_item	a GtkMenuItem

Returns

`TRUE` if an embedded underline in the label indicates the mnemonic accelerator key.

Since: 2.16

---

gtk_menu_item_set_use_underline ()

```
void
gtk_menu_item_set_use_underline (GtkMenuItem *menu_item,
                                 gboolean setting);
```

If true, an underline in the text indicates the next character should be used for the mnemonic accelerator key.

Parameters

menu_item	a GtkMenuItem
setting	`TRUE` if underlines in the text indicate mnemonics

Since: 2.16

---

gtk_menu_item_set_submenu ()

```
void
gtk_menu_item_set_submenu (GtkMenuItem *menu_item,
                           GtkWidget *submenu);
```

Sets or replaces the menu item's submenu, or removes it when a `NULL` submenu is passed.

Parameters

menu_item	a GtkMenuItem	
submenu	the submenu, or `NULL`.	[allow-none][type Gtk.Menu]

---

gtk_menu_item_get_submenu ()

```
GtkWidget *
gtk_menu_item_get_submenu (GtkMenuItem *menu_item);
```

Gets the submenu underneath this menu item, if any. See `gtk_menu_item_set_submenu()`.

Parameters

menu_item	a GtkMenuItem

Returns

submenu for this menu item, or `NULL` if none.

[nullable][transfer none]

---

gtk_menu_item_set_accel_path ()

```
void
gtk_menu_item_set_accel_path (GtkMenuItem *menu_item,
                              const gchar *accel_path);
```

Set the accelerator path on `menu_item` , through which runtime changes of the menu item's accelerator caused by the user can be identified and saved to persistent storage (see `gtk_accel_map_save()` on this). To set up a default accelerator for this menu item, call `gtk_accel_map_add_entry()` with the same `accel_path` . See also `gtk_accel_map_add_entry()` on the specifics of accelerator paths, and `gtk_menu_set_accel_path()` for a more convenient variant of this function.

This function is basically a convenience wrapper that handles calling `gtk_widget_set_accel_path()` with the appropriate accelerator group for the menu item.

Note that you do need to set an accelerator on the parent menu with `gtk_menu_set_accel_group()` for this to work.

Note that `accel_path` string will be stored in a GQuark. Therefore, if you pass a static string, you can save some memory by interning it first with `g_intern_static_string()`.

Parameters

menu_item	a valid GtkMenuItem	
accel_path	accelerator path, corresponding to this menu item's functionality, or `NULL` to unset the current path.	[allow-none]

---

gtk_menu_item_get_accel_path ()

```
const gchar *
gtk_menu_item_get_accel_path (GtkMenuItem *menu_item);
```

Retrieve the accelerator path that was previously set on `menu_item`.

See `gtk_menu_item_set_accel_path()` for details.

Parameters

menu_item	a valid GtkMenuItem

Returns

the accelerator path corresponding to this menu item's functionality, or `NULL` if not set.

[nullable][transfer none]

Since: 2.14

---

gtk_menu_item_select ()

```
void
gtk_menu_item_select (GtkMenuItem *menu_item);
```

Emits the "select" signal on the given item.

Parameters

menu_item	the menu item

---

gtk_menu_item_deselect ()

```
void
gtk_menu_item_deselect (GtkMenuItem *menu_item);
```

Emits the "deselect" signal on the given item.

Parameters

menu_item	the menu item

---

gtk_menu_item_activate ()

```
void
gtk_menu_item_activate (GtkMenuItem *menu_item);
```

Emits the “activate” signal on the given item

Parameters

menu_item	the menu item

---

gtk_menu_item_toggle_size_request ()

```
void
gtk_menu_item_toggle_size_request (GtkMenuItem *menu_item,
                                   gint *requisition);
```

Emits the “toggle-size-request” signal on the given item.

Parameters

menu_item	the menu item
requisition	the requisition to use as signal data. [inout]

---

gtk_menu_item_toggle_size_allocate ()

```
void
gtk_menu_item_toggle_size_allocate (GtkMenuItem *menu_item,
                                    gint allocation);
```

Emits the “toggle-size-allocate” signal on the given item.

Parameters

menu_item	the menu item.
allocation	the allocation to use as signal data.

---

gtk_menu_item_get_reserve_indicator ()

```
gboolean
gtk_menu_item_get_reserve_indicator (GtkMenuItem *menu_item);
```

Returns whether the `menu_item` reserves space for the submenu indicator, regardless if it has a submenu or not.

Parameters

menu_item	a GtkMenuItem

Returns

`TRUE` if `menu_item` always reserves space for the submenu indicator

Since: 3.0

---

gtk_menu_item_set_reserve_indicator ()

```
void
gtk_menu_item_set_reserve_indicator (GtkMenuItem *menu_item,
                                     gboolean reserve);
```

Sets whether the `menu_item` should reserve space for the submenu indicator, regardless if it actually has a submenu or not.

There should be little need for applications to call this functions.

Parameters

menu_item	a GtkMenuItem
reserve	the new value

Since: 3.0

Types and Values

struct GtkMenuItem

```
struct GtkMenuItem;
```

---

struct GtkMenuItemClass

```
struct GtkMenuItemClass {
  GtkBinClass parent_class;


  /* If the following flag is true, then we should always
   * hide the menu when the MenuItem is activated. Otherwise,
   * it is up to the caller. For instance, when navigating
   * a menu with the keyboard, <Space> doesn't hide, but
   * <Return> does.
   */
  guint hide_on_activate : 1;
```

```
  void (* activate)             (GtkMenuItem *menu_item);
  void (* activate_item)        (GtkMenuItem *menu_item);
  void (* toggle_size_request)  (GtkMenuItem *menu_item,
                                 gint        *requisition);
  void (* toggle_size_allocate) (GtkMenuItem *menu_item,
                                 gint         allocation);
  void (* set_label)            (GtkMenuItem *menu_item,
                                 const gchar *label);
  const gchar * (* get_label)   (GtkMenuItem *menu_item);

  void (* select)               (GtkMenuItem *menu_item);
  void (* deselect)             (GtkMenuItem *menu_item);
};
```

Members

guint `hide_on_activate` : 1;	If `TRUE`, then we should always hide the menu when the `GtkMenuItem` is activated. Otherwise, it is up to the caller.
`activate` ()	Signal emitted when the item is activated.
`activate_item` ()	Signal emitted when the item is activated, but also if the menu item has a submenu.
`toggle_size_request` ()	
`toggle_size_allocate` ()	
`set_label` ()	Sets `text` on the GtkMenuItem label
`get_label` ()	Gets `text` from the GtkMenuItem label
`select` ()	Signal emitted when the item is selected.
`deselect` ()	Signal emitted when the item is deselected.

Property Details

The `"accel-path"` property

```
  "accel-path"               gchar *
```

Sets the accelerator path of the menu item, through which runtime changes of the menu item's accelerator caused by the user can be identified and saved to persistant storage.

Flags: Read / Write

Default value: NULL

Since: 2.14

The “label” property

“label” gchar *

The text for the child label.

Flags: Read / Write

Default value: ""

Since: 2.16

The “right-justified” property

“right-justified” gboolean

Sets whether the menu item appears justified at the right side of a menu bar.

Flags: Read / Write

Default value: FALSE

Since: 2.14

The “submenu” property

“submenu” GtkMenu *

The submenu attached to the menu item, or NULL if it has none.

Flags: Read / Write

Since: 2.12

The “use-underline” property

“use-underline” gboolean

TRUE if underlines in the text indicate mnemonics.

Flags: Read / Write

Default value: FALSE

Since: 2.16

Style Property Details

The “arrow-scaling” style property

“arrow-scaling” gfloat

Amount of space used up by the arrow, relative to the menu item's font size.

GtkMenuItem:arrow-scaling has been deprecated since version 3.20 and should not be used in newly-written code.

use the standard min-width/min-height CSS properties on the arrow node; the value of this style property is ignored.

Flags: Read

Allowed values: [0,2]

Default value: 0.8

The “arrow-spacing” style property

“arrow-spacing” gint

Spacing between menu item label and submenu arrow.

GtkMenuItem:arrow-spacing has been deprecated since version 3.20 and should not be used in newly-written code.

use the standard margin CSS property on the arrow node; the value of this style property is ignored.

Flags: Read

Allowed values: >= 0

Default value: 10

The “horizontal-padding” style property

“horizontal-padding” gint

Padding to left and right of the menu item.

GtkMenuItem:horizontal-padding has been deprecated since version 3.8 and should not be used in newly-written code.

use the standard padding CSS property (through objects like GtkStyleContext and GtkCssProvider); the value of this style property is ignored.

Flags: Read

Allowed values: >= 0

Default value: 0

The “selected-shadow-type” style property

“selected-shadow-type” GtkShadowType

The shadow type when the item is selected.

GtkMenuItem:selected-shadow-type has been deprecated since version 3.20 and should not be used in newly-written code.

Use CSS to determine the shadow; the value of this style property is ignored.

Flags: Read

Default value: GTK_SHADOW_NONE

The “toggle-spacing” style property

```
  “toggle-spacing”           gint
```

Spacing between menu icon and label.

`GtkMenuItem:toggle-spacing` has been deprecated since version 3.20 and should not be used in newly-written code.

use the standard margin CSS property on the check or radio nodes; the value of this style property is ignored.

Flags: Read

Allowed values: >= 0

Default value: 5

The “width-chars” style property

```
  “width-chars”              gint
```

The minimum desired width of the menu item in characters.

`GtkMenuItem:width-chars` has been deprecated since version 3.20 and should not be used in newly-written code.

Use the standard CSS property min-width; the value of this style property is ignored.

Flags: Read

Allowed values: >= 0

Default value: 12

Since: 2.14

Signal Details

The “activate” signal

```
void
user_function (GtkMenuItem *menuitem,
               gpointer     user_data)
```

Emitted when the item is activated.

Parameters

menuitem	the object which received the signal.
user_data	user data set when the signal handler was connected.

Flags: Action

The `"activate-item"` signal

```
void
user_function (GtkMenuItem *menuitem,
               gpointer     user_data)
```

Emitted when the item is activated, but also if the menu item has a submenu. For normal applications, the relevant signal is "activate".

Parameters

menuitem	the object which received the signal.
user_data	user data set when the signal handler was connected.

Flags: Run First

The `"deselect"` signal

```
void
user_function (GtkMenuItem *menuitem,
               gpointer     user_data)
```

Flags: Run First

The `"select"` signal

```
void
user_function (GtkMenuItem *menuitem,
               gpointer     user_data)
```

Flags: Run First

The `"toggle-size-allocate"` signal

```
void
user_function (GtkMenuItem *menuitem,
               gint         arg1,
               gpointer     user_data)
```

Flags: Run First

The "toggle-size-request" signal

```
void
user_function (GtkMenuItem *menuitem,
               gpointer     arg1,
               gpointer     user_data)
```

Flags: Run First

See Also

GtkBin, GtkMenuShell

GtkRadioMenuItem

GtkRadioMenuItem — A choice from multiple check menu items

Functions

GtkWidget *	gtk_radio_menu_item_new ()
GtkWidget *	gtk_radio_menu_item_new_with_label ()
GtkWidget *	gtk_radio_menu_item_new_with_mnemonic ()
GtkWidget *	gtk_radio_menu_item_new_from_widget ()
GtkWidget *	gtk_radio_menu_item_new_with_label_from_widget ()
GtkWidget *	gtk_radio_menu_item_new_with_mnemonic_from_widget ()
void	gtk_radio_menu_item_set_group ()
GSList *	gtk_radio_menu_item_get_group ()
void	gtk_radio_menu_item_join_group ()

Properties

GtkRadioMenuItem *	group	Write

Signals

void	group-changed	Run First

Types and Values

struct	GtkRadioMenuItem

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
```

```
└── GtkContainer
    └── GtkBin
        └── GtkMenuItem
            └── GtkCheckMenuItem
                └── GtkRadioMenuItem
```

Implemented Interfaces

GtkRadioMenuItem implements AtkImplementorIface, GtkBuildable, GtkActivatable and GtkActionable.

Includes

```
#include <gtk/gtk.h>
```

Description

A radio menu item is a check menu item that belongs to a group. At each instant exactly one of the radio menu items from a group is selected.

The group list does not need to be freed, as each GtkRadioMenuItem will remove itself and its list item when it is destroyed.

The correct way to create a group of radio menu items is approximatively this:

How to create a group of radio menu items.

```
GSList *group = NULL;
GtkWidget *item;
gint i;

for (i = 0; i < 5; i++)
{
  item = gtk_radio_menu_item_new_with_label
(group, "This is an example");
  group = gtk_radio_menu_item_get_group
(GTK_RADIO_MENU_ITEM (item));
  if (i == 1)
    gtk_check_menu_item_set_active
(GTK_CHECK_MENU_ITEM (item), TRUE);
}
```

CSS nodes

```
menuitem
├── radio.left
└── <child>
```

GtkRadioMenuItem has a main CSS node with name menuitem, and a subnode with name radio, which gets the .left or .right style class.

Functions

gtk_radio_menu_item_new ()

```
GtkWidget *
gtk_radio_menu_item_new (GSList *group);
```

Creates a new GtkRadioMenuItem.

Parameters

group	the group to which the radio menu item is to be attached, or NULL.	[element-type GtkRadioMenuItem] [allow-none]

Returns

a new GtkRadioMenuItem

---

gtk_radio_menu_item_new_with_label ()

```
GtkWidget *
gtk_radio_menu_item_new_with_label (GSList *group,
                                    const gchar *label);
```

Creates a new GtkRadioMenuItem whose child is a simple GtkLabel.

Parameters

group	group the radio menu item is inside, or NULL.	[element-type GtkRadioMenuItem] [allow-none]
label	the text for the label	

Returns

A new GtkRadioMenuItem.

[transfer none]

---

gtk_radio_menu_item_new_with_mnemonic ()

```
GtkWidget *
gtk_radio_menu_item_new_with_mnemonic (GSList *group,
                                       const gchar *label);
```

Creates a new GtkRadioMenuItem containing a label. The label will be created using gtk_label_new_with_mnemonic(), so underscores in label indicate the mnemonic for the menu item.

Parameters

group	group the radio menu item is inside, or NULL.	[element-type GtkRadioMenuItem] [allow-none]

label	the text of the button, with an underscore in front of the mnemonic character

Returns

a new GtkRadioMenuItem

---

gtk_radio_menu_item_new_from_widget ()

```
GtkWidget *
gtk_radio_menu_item_new_from_widget (GtkRadioMenuItem *group);
```

Creates a new GtkRadioMenuItem adding it to the same group as `group` .

[constructor]

Parameters

group	An existing GtkRadioMenuItem.	[allow-none]

Returns

The new GtkRadioMenuItem.

[transfer none]

Since: 2.4

---

gtk_radio_menu_item_new_with_label_from_widget ()

```
GtkWidget *
gtk_radio_menu_item_new_with_label_from_widget
                               (GtkRadioMenuItem *group,
                                const gchar *label);
```

Creates a new GtkRadioMenuItem whose child is a simple GtkLabel. The new GtkRadioMenuItem is added to the same group as `group` .

[constructor]

Parameters

group	an existing GtkRadioMenuItem.	[allow-none]
label	the text for the label.	[allow-none]

Returns

The new GtkRadioMenuItem.

[transfer none]

Since: 2.4

---

gtk_radio_menu_item_new_with_mnemonic_from_widget ()

```
GtkWidget *
gtk_radio_menu_item_new_with_mnemonic_from_widget
                                (GtkRadioMenuItem *group,
                                 const gchar *label);
```

Creates a new GtkRadioMenuItem containing a label. The label will be created using `gtk_label_new_with_mnemonic()`, so underscores in label indicate the mnemonic for the menu item.

The new GtkRadioMenuItem is added to the same group as `group` .

[constructor]

Parameters

group	An existing GtkRadioMenuItem.	[allow-none]
label	the text of the button, with an underscore in front of the mnemonic character.	[allow-none]

Returns

The new GtkRadioMenuItem.

[transfer none]

Since: 2.4

---

gtk_radio_menu_item_set_group ()

```
void
gtk_radio_menu_item_set_group (GtkRadioMenuItem *radio_menu_item,
                               GSList *group);
```

Sets the group of a radio menu item, or changes it.

Parameters

radio_menu_item	a GtkRadioMenuItem.	
group	the new group, or `NULL`.	[element-type GtkRadioMenuItem] [allow-none]

---

gtk_radio_menu_item_get_group ()

```
GSList *
gtk_radio_menu_item_get_group (GtkRadioMenuItem *radio_menu_item);
```

Returns the group to which the radio menu item belongs, as a GList of GtkRadioMenuItem. The list belongs to GTK+ and should not be freed.

Parameters

radio_menu_item	a GtkRadioMenuItem

Returns

the group of `radio_menu_item`.

[element-type GtkRadioMenuItem][transfer none]

---

gtk_radio_menu_item_join_group ()

```
void
gtk_radio_menu_item_join_group (GtkRadioMenuItem *radio_menu_item,
                                GtkRadioMenuItem *group_source);
```

Joins a GtkRadioMenuItem object to the group of another GtkRadioMenuItem object.

This function should be used by language bindings to avoid the memory manangement of the opaque GSList of `gtk_radio_menu_item_get_group()` and `gtk_radio_menu_item_set_group()`.

A common way to set up a group of GtkRadioMenuItem instances is:

```
GtkRadioMenuItem *last_item = NULL;

while ( ...more items to add... )
  {
    GtkRadioMenuItem *radio_item;

    radio_item = gtk_radio_menu_item_new
(...);

    gtk_radio_menu_item_join_group
(radio_item, last_item);
    last_item = radio_item;
  }
```

Parameters

radio_menu_item	a GtkRadioMenuItem	
group_source	a GtkRadioMenuItem whose group we are joining, or `NULL` to remove the `radio_menu_item` from its current group.	[allow-none]

Since: 3.18

Types and Values

struct GtkRadioMenuItem

```
struct GtkRadioMenuItem;
```

Property Details

The “group” property

```
  “group”                    GtkRadioMenuItem *
```

The radio menu item whose group this widget belongs to.

Flags: Write

Since: 2.8

Signal Details

The “group-changed” signal

```
void
user_function (GtkRadioMenuItem *radiomenuitem,
               gpointer          user_data)
```

Flags: Run First

See Also

GtkMenuItem, GtkCheckMenuItem

GtkCheckMenuItem

GtkCheckMenuItem — A menu item with a check box

Functions

GtkWidget *	gtk_check_menu_item_new ()
GtkWidget *	gtk_check_menu_item_new_with_label ()
GtkWidget *	gtk_check_menu_item_new_with_mnemonic ()
gboolean	gtk_check_menu_item_get_active ()
void	gtk_check_menu_item_set_active ()
void	gtk_check_menu_item_toggled ()
gboolean	gtk_check_menu_item_get_inconsistent ()
void	gtk_check_menu_item_set_inconsistent ()

void	gtk_check_menu_item_set_draw_as_radio ()
gboolean	gtk_check_menu_item_get_draw_as_radio ()

Properties

gboolean	active	Read / Write
gboolean	draw-as-radio	Read / Write
gboolean	inconsistent	Read / Write

Style Properties

gint	indicator-size	Read

Signals

void	toggled	Run First

Types and Values

struct	GtkCheckMenuItem
struct	GtkCheckMenuItemClass

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkContainer
            └── GtkBin
                └── GtkMenuItem
                    └── GtkCheckMenuItem
                        └── GtkRadioMenuItem
```

Implemented Interfaces

GtkCheckMenuItem implements AtkImplementorIface, GtkBuildable, GtkActivatable and GtkActionable.

Includes

```
#include <gtk/gtk.h>
```

Description

A GtkCheckMenuItem is a menu item that maintains the state of a boolean value in addition to a GtkMenuItem usual role in activating application code.

A check box indicating the state of the boolean value is displayed at the left side of the GtkMenuItem. Activating the GtkMenuItem toggles the value.

CSS nodes

```
1                                                   menuitem
2                                                   ├── check.left
3                                                   └── <child>
```

GtkCheckMenuItem has a main CSS node with name menuitem, and a subnode with name check, which gets the .left or .right style class.

Functions

gtk_check_menu_item_new ()

```
GtkWidget *
gtk_check_menu_item_new (void);
```

Creates a new GtkCheckMenuItem.

Returns

a new GtkCheckMenuItem.

---

gtk_check_menu_item_new_with_label ()

```
GtkWidget *
gtk_check_menu_item_new_with_label (const gchar *label);
```

Creates a new GtkCheckMenuItem with a label.

Parameters

label	the string to use for the label.

Returns

a new GtkCheckMenuItem.

---

gtk_check_menu_item_new_with_mnemonic ()

```
GtkWidget *
gtk_check_menu_item_new_with_mnemonic (const gchar *label);
```

Creates a new GtkCheckMenuItem containing a label. The label will be created using `gtk_label_new_with_mnemonic()`, so underscores in `label` indicate the mnemonic for the menu item.

Parameters

label	The text of the button, with an

underscore in front of the character

Returns

a new GtkCheckMenuItem

---

gtk_check_menu_item_get_active ()

```
gboolean
gtk_check_menu_item_get_active (GtkCheckMenuItem *check_menu_item);
```

Returns whether the check menu item is active. See `gtk_check_menu_item_set_active()`.

Parameters

check_menu_item	a GtkCheckMenuItem

Returns

`TRUE` if the menu item is checked.

---

gtk_check_menu_item_set_active ()

```
void
gtk_check_menu_item_set_active (GtkCheckMenuItem *check_menu_item,
                                gboolean is_active);
```

Sets the active state of the menu item's check box.

Parameters

check_menu_item	a GtkCheckMenuItem.
is_active	boolean value indicating whether the check box is active.

---

gtk_check_menu_item_toggled ()

```
void
gtk_check_menu_item_toggled (GtkCheckMenuItem *check_menu_item);
```

Emits the "toggled" signal.

Parameters

check_menu_item	a GtkCheckMenuItem.

---

gtk_check_menu_item_get_inconsistent ()

```
gboolean
gtk_check_menu_item_get_inconsistent (GtkCheckMenuItem *check_menu_item);
```

Retrieves the value set by `gtk_check_menu_item_set_inconsistent()`.

Parameters

check_menu_item	a GtkCheckMenuItem

Returns

`TRUE` if inconsistent

---

gtk_check_menu_item_set_inconsistent ()

```
void
gtk_check_menu_item_set_inconsistent (GtkCheckMenuItem *check_menu_item,
                                      gboolean setting);
```

If the user has selected a range of elements (such as some text or spreadsheet cells) that are affected by a boolean setting, and the current values in that range are inconsistent, you may want to display the check in an “in between” state. This function turns on “in between” display. Normally you would turn off the inconsistent state again if the user explicitly selects a setting. This has to be done manually, `gtk_check_menu_item_set_inconsistent()` only affects visual appearance, it doesn’t affect the semantics of the widget.

Parameters

check_menu_item	a GtkCheckMenuItem
setting	`TRUE` to display an “inconsistent” third state check

---

gtk_check_menu_item_set_draw_as_radio ()

```
void
gtk_check_menu_item_set_draw_as_radio (GtkCheckMenuItem *check_menu_item,
                                       gboolean draw_as_radio);
```

Sets whether `check_menu_item` is drawn like a GtkRadioMenuItem

Parameters

check_menu_item	a GtkCheckMenuItem
draw_as_radio	whether `check_menu_item` is drawn like a GtkRadioMenuItem

Since: 2.4

---

gtk_check_menu_item_get_draw_as_radio ()

```
gboolean
gtk_check_menu_item_get_draw_as_radio (GtkCheckMenuItem *check_menu_item);
```

Returns whether `check_menu_item` looks like a GtkRadioMenuItem

Parameters

check_menu_item	a GtkCheckMenuItem

Returns

Whether `check_menu_item` looks like a GtkRadioMenuItem

Since: 2.4

Types and Values

struct GtkCheckMenuItem

```
struct GtkCheckMenuItem;
```

---

struct GtkCheckMenuItemClass

```
struct GtkCheckMenuItemClass {
  GtkMenuItemClass parent_class;


  void (* toggled)        (GtkCheckMenuItem *check_menu_item);
  void (* draw_indicator) (GtkCheckMenuItem *check_menu_item,
                           cairo_t          *cr);
};
```

Members

`toggled` ()	Signal emitted when the state of the check box is changed.
`draw_indicator` ()	Called to draw the check indicator.

Property Details

The “active” property

“active” gboolean

Whether the menu item is checked.

Flags: Read / Write

Default value: FALSE

The “draw-as-radio” property

“draw-as-radio” gboolean

Whether the menu item looks like a radio menu item.

Flags: Read / Write

Default value: FALSE

The “inconsistent” property

“inconsistent” gboolean

Whether to display an "inconsistent" state.

Flags: Read / Write

Default value: FALSE

Style Property Details

The “indicator-size” style property

“indicator-size” gint

The size of the check or radio indicator.

`GtkCheckMenuItem:indicator-size` has been deprecated since version 3.20 and should not be used in newly-written code.

Use the standard CSS property min-width on the check or radio nodes; the value of this style property is ignored.

Flags: Read

Allowed values: >= 0

Default value: 16

Signal Details

The "toggled" signal

```
void
user_function (GtkCheckMenuItem *checkmenuitem,
               gpointer          user_data)
```

This signal is emitted when the state of the check box is changed.

A signal handler can use gtk_check_menu_item_get_active() to discover the new state.

Parameters

checkmenuitem	the object which received the signal.
user_data	user data set when the signal handler was connected.

Flags: Run First

GtkSeparatorMenuItem

GtkSeparatorMenuItem — A separator used in menus

Functions

GtkWidget *	gtk_separator_menu_item_new ()

Types and Values

struct	GtkSeparatorMenuItem
struct	GtkSeparatorMenuItemClass

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkContainer
            └── GtkBin
                └── GtkMenuItem
                    └── GtkSeparatorMenuItem
```

Implemented Interfaces

GtkSeparatorMenuItem implements AtkImplementorIface, GtkBuildable, GtkActivatable and GtkActionable.

Includes

```
#include <gtk/gtk.h>
```

Description

The GtkSeparatorMenuItem is a separator used to group items within a menu. It displays a horizontal line with a shadow to make it appear sunken into the interface.

CSS nodes

GtkSeparatorMenuItem has a single CSS node with name separator.

Functions

gtk_separator_menu_item_new ()

```
GtkWidget *
gtk_separator_menu_item_new (void);
```

Creates a new GtkSeparatorMenuItem.

Returns

a new GtkSeparatorMenuItem.

Types and Values

struct GtkSeparatorMenuItem

```
struct GtkSeparatorMenuItem;
```

---

struct GtkSeparatorMenuItemClass

```
struct GtkSeparatorMenuItemClass {
  GtkMenuItemClass parent_class;
};
```

Members

GtkToolShell

GtkToolShell — Interface for containers containing GtkToolItem widgets

Functions

PangoEllipsizeMode	gtk_tool_shell_get_ellipsize_mode ()
GtkIconSize	gtk_tool_shell_get_icon_size ()
GtkOrientation	gtk_tool_shell_get_orientation ()
GtkReliefStyle	gtk_tool_shell_get_relief_style ()
GtkToolbarStyle	gtk_tool_shell_get_style ()
gfloat	gtk_tool_shell_get_text_alignment ()
GtkOrientation	gtk_tool_shell_get_text_orientation ()
void	gtk_tool_shell_rebuild_menu ()
GtkSizeGroup *	gtk_tool_shell_get_text_size_group ()

Types and Values

	GtkToolShell
struct	GtkToolShellIface

Object Hierarchy

```
GInterface
╰── GtkToolShell
```

Prerequisites

GtkToolShell requires GtkWidget.

Known Implementations

GtkToolShell is implemented by GtkToolItemGroup and GtkToolbar.

Includes

```
#include <gtk/gtk.h>
```

Description

The GtkToolShell interface allows container widgets to provide additional information when embedding GtkToolItem widgets.

Functions

gtk_tool_shell_get_ellipsize_mode ()

```
PangoEllipsizeMode
gtk_tool_shell_get_ellipsize_mode (GtkToolShell *shell);
```

Retrieves the current ellipsize mode for the tool shell. Tool items must not call this function directly, but rely on

gtk_tool_item_get_ellipsize_mode() instead.

Parameters

shell	a GtkToolShell

Returns

the current ellipsize mode of `shell`

Since: 2.20

---

gtk_tool_shell_get_icon_size ()

```
GtkIconSize
gtk_tool_shell_get_icon_size (GtkToolShell *shell);
```

Retrieves the icon size for the tool shell. Tool items must not call this function directly, but rely on gtk_tool_item_get_icon_size() instead.

Parameters

shell	a GtkToolShell

Returns

the current size (GtkIconSize) for icons of `shell` .

[type int]

Since: 2.14

---

gtk_tool_shell_get_orientation ()

```
GtkOrientation
gtk_tool_shell_get_orientation (GtkToolShell *shell);
```

Retrieves the current orientation for the tool shell. Tool items must not call this function directly, but rely on gtk_tool_item_get_orientation() instead.

Parameters

shell	a GtkToolShell

Returns

the current orientation of `shell`

Since: 2.14

---

gtk_tool_shell_get_relief_style ()

```
GtkReliefStyle
gtk_tool_shell_get_relief_style (GtkToolShell *shell);
```

Returns the relief style of buttons on `shell` . Tool items must not call this function directly, but rely on `gtk_tool_item_get_relief_style()` instead.

Parameters

shell	a GtkToolShell

Returns

The relief style of buttons on `shell` .

Since: 2.14

---

gtk_tool_shell_get_style ()

```
GtkToolbarStyle
gtk_tool_shell_get_style (GtkToolShell *shell);
```

Retrieves whether the tool shell has text, icons, or both. Tool items must not call this function directly, but rely on `gtk_tool_item_get_toolbar_style()` instead.

Parameters

shell	a GtkToolShell

Returns

the current style of `shell`

Since: 2.14

---

gtk_tool_shell_get_text_alignment ()

```
gfloat
gtk_tool_shell_get_text_alignment (GtkToolShell *shell);
```

Retrieves the current text alignment for the tool shell. Tool items must not call this function directly, but rely on `gtk_tool_item_get_text_alignment()` instead.

Parameters

shell	a GtkToolShell

Returns

the current text alignment of `shell`

Since: 2.20

---

gtk_tool_shell_get_text_orientation ()

```
GtkOrientation
gtk_tool_shell_get_text_orientation (GtkToolShell *shell);
```

Retrieves the current text orientation for the tool shell. Tool items must not call this function directly, but rely on `gtk_tool_item_get_text_orientation()` instead.

Parameters

shell	a GtkToolShell

Returns

the current text orientation of `shell`

Since: 2.20

---

gtk_tool_shell_rebuild_menu ()

```
void
gtk_tool_shell_rebuild_menu (GtkToolShell *shell);
```

Calling this function signals the tool shell that the overflow menu item for tool items have changed. If there is an overflow menu and if it is visible when this function it called, the menu will be rebuilt.

Tool items must not call this function directly, but rely on `gtk_tool_item_rebuild_menu()` instead.

Parameters

shell	a GtkToolShell

Since: 2.14

---

gtk_tool_shell_get_text_size_group ()

```
GtkSizeGroup *
gtk_tool_shell_get_text_size_group (GtkToolShell *shell);
```

Retrieves the current text size group for the tool shell. Tool items must not call this function directly, but rely on

gtk_tool_item_get_text_size_group() instead.

Parameters

shell	a GtkToolShell

Returns

the current text size group of `shell`.

[transfer none]

Since: 2.20

Types and Values

GtkToolShell

```
typedef struct _GtkToolShell GtkToolShell;
```

Dummy structure for accessing instances of GtkToolShellIface.

---

struct GtkToolShellIface

```
struct GtkToolShellIface {
  GtkIconSize        (*get_icon_size)        (GtkToolShell *shell);
  GtkOrientation     (*get_orientation)      (GtkToolShell *shell);
  GtkToolbarStyle    (*get_style)            (GtkToolShell *shell);
  GtkReliefStyle     (*get_relief_style)     (GtkToolShell *shell);
  void               (*rebuild_menu)         (GtkToolShell *shell);
  GtkOrientation     (*get_text_orientation) (GtkToolShell *shell);
  gfloat             (*get_text_alignment)   (GtkToolShell *shell);
  PangoEllipsizeMode (*get_ellipsize_mode)   (GtkToolShell *shell);
  GtkSizeGroup *     (*get_text_size_group)  (GtkToolShell *shell);
};
```

Virtual function table for the GtkToolShell interface.

Members

`get_icon_size` ()	mandatory implementation of gtk_tool_shell_get_icon_size().
`get_orientation` ()	mandatory implementation of gtk_tool_shell_get_orientation().
`get_style` ()	mandatory implementation of gtk_tool_shell_get_style().
`get_relief_style` ()	optional implementation of gtk_tool_shell_get_relief_sty

	le().
rebuild_menu ()	optional implementation of gtk_tool_shell_rebuild_menu().
get_text_orientation ()	optional implementation of gtk_tool_shell_get_text_orientation().
get_text_alignment ()	optional implementation of gtk_tool_shell_get_text_alignment().
get_ellipsize_mode ()	optional implementation of gtk_tool_shell_get_ellipsize_mode().
get_text_size_group ()	optional implementation of gtk_tool_shell_get_text_size_group().

See Also

GtkToolbar, GtkToolItem

GtkToolbar

GtkToolbar — Create bars of buttons and other widgets

Functions

GtkWidget *	gtk_toolbar_new ()
void	gtk_toolbar_insert ()
gint	gtk_toolbar_get_item_index ()
gint	gtk_toolbar_get_n_items ()
GtkToolItem *	gtk_toolbar_get_nth_item ()
gint	gtk_toolbar_get_drop_index ()
void	gtk_toolbar_set_drop_highlight_item ()
void	gtk_toolbar_set_show_arrow ()
void	gtk_toolbar_unset_icon_size ()
gboolean	gtk_toolbar_get_show_arrow ()
GtkToolbarStyle	gtk_toolbar_get_style ()
GtkIconSize	gtk_toolbar_get_icon_size ()
GtkReliefStyle	gtk_toolbar_get_relief_style ()
void	gtk_toolbar_set_style ()
void	gtk_toolbar_set_icon_size ()
void	gtk_toolbar_unset_style ()

Properties

GtkIconSize	icon-size	Read / Write
gboolean	icon-size-set	Read / Write
gboolean	show-arrow	Read / Write
GtkToolbarStyle	toolbar-style	Read / Write

Child Properties

gboolean	expand	Read / Write
gboolean	homogeneous	Read / Write

Style Properties

GtkReliefStyle	button-relief	Read
gint	internal-padding	Read
gint	max-child-expand	Read
GtkShadowType	shadow-type	Read
gint	space-size	Read
GtkToolbarSpaceStyle	space-style	Read

Signals

gboolean	focus-home-or-end	Action
void	orientation-changed	Run First
gboolean	popup-context-menu	Run Last
void	style-changed	Run First

Types and Values

struct	GtkToolbar
enum	GtkToolbarSpaceStyle

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkContainer
            └── GtkToolbar
```

Implemented Interfaces

GtkToolbar implements AtkImplementorIface, GtkBuildable, GtkToolShell and GtkOrientable.

Includes

```
#include <gtk/gtk.h>
```

Description

A toolbar is created with a call to `gtk_toolbar_new()`.

A toolbar can contain instances of a subclass of GtkToolItem. To add a GtkToolItem to the a toolbar, use `gtk_toolbar_insert()`. To remove an item from the toolbar use `gtk_container_remove()`. To add a button to the toolbar, add an instance of GtkToolButton.

Toolbar items can be visually grouped by adding instances of GtkSeparatorToolItem to the toolbar. If the GtkToolbar child property "expand" is TRUE and the property "draw" is set to FALSE, the effect is to force all following items to the end of the toolbar.

By default, a toolbar can be shrunk, upon which it will add an arrow button to show an overflow menu offering access to any GtkToolItem child that has a proxy menu item. To disable this and request enough size for all children, call `gtk_toolbar_set_show_arrow()` to set "show-arrow" to `FALSE`.

Creating a context menu for the toolbar can be done by connecting to the "popup-context-menu" signal.

CSS nodes

GtkToolbar has a single CSS node with name toolbar.

Functions

gtk_toolbar_new ()

```
GtkWidget *
gtk_toolbar_new (void);
```

Creates a new toolbar.

Returns

the newly-created toolbar.

---

gtk_toolbar_insert ()

```
void
gtk_toolbar_insert (GtkToolbar *toolbar,
                    GtkToolItem *item,
                    gint pos);
```

Insert a GtkToolItem into the toolbar at position `pos` . If `pos` is 0 the item is prepended to the start of the toolbar. If `pos` is negative, the item is appended to the end of the toolbar.

Parameters

toolbar	a GtkToolbar
item	a GtkToolItem

pos	the position of the new item

Since: 2.4

---

gtk_toolbar_get_item_index ()

```
gint
gtk_toolbar_get_item_index (GtkToolbar *toolbar,
                            GtkToolItem *item);
```

Returns the position of `item` on the toolbar, starting from 0. It is an error if `item` is not a child of the toolbar.

Parameters

toolbar	a GtkToolbar
item	a GtkToolItem that is a child of `toolbar`

Returns

the position of item on the toolbar.

Since: 2.4

---

gtk_toolbar_get_n_items ()

```
gint
gtk_toolbar_get_n_items (GtkToolbar *toolbar);
```

Returns the number of items on the toolbar.

Parameters

toolbar	a GtkToolbar

Returns

the number of items on the toolbar

Since: 2.4

---

gtk_toolbar_get_nth_item ()

```
GtkToolItem *
gtk_toolbar_get_nth_item (GtkToolbar *toolbar,
                          gint n);
```

Returns the `n` 'th item on `toolbar` , or `NULL` if the toolbar does not contain an `n` 'th item.

Parameters

toolbar	a GtkToolbar
n	A position on the toolbar

Returns

The `n` 'th GtkToolItem on `toolbar` , or `NULL` if there isn’t an `n` 'th item.

[nullable][transfer none]

Since: 2.4

---

gtk_toolbar_get_drop_index ()

```
gint
gtk_toolbar_get_drop_index (GtkToolbar *toolbar,
                            gint x,
                            gint y);
```

Returns the position corresponding to the indicated point on `toolbar` . This is useful when dragging items to the toolbar: this function returns the position a new item should be inserted.

`x` and `y` are in `toolbar` coordinates.

Parameters

toolbar	a GtkToolbar
x	x coordinate of a point on the toolbar
y	y coordinate of a point on the toolbar

Returns

The position corresponding to the point (`x` , `y`) on the toolbar.

Since: 2.4

---

gtk_toolbar_set_drop_highlight_item ()

```
void
gtk_toolbar_set_drop_highlight_item (GtkToolbar *toolbar,
                                     GtkToolItem *tool_item,
                                     gint index_);
```

Highlights `toolbar` to give an idea of what it would look like if `item` was added to `toolbar` at the position indicated by `index_` . If `item` is `NULL`, highlighting is turned off. In that case `index_` is ignored.

The `tool_item` passed to this function must not be part of any widget hierarchy. When an item is set as drop highlight item it can not added to any widget hierarchy or used as highlight item for another toolbar.

Parameters

toolbar	a GtkToolbar	
tool_item	a GtkToolItem, or `NULL` to turn of highlighting.	[allow-none]
index_	a position on `toolbar`	

Since: 2.4

---

gtk_toolbar_set_show_arrow ()

```
void
gtk_toolbar_set_show_arrow (GtkToolbar *toolbar,
                            gboolean show_arrow);
```

Sets whether to show an overflow menu when `toolbar` isn't allocated enough size to show all of its items. If `TRUE`, items which can't fit in `toolbar` , and which have a proxy menu item set by `gtk_tool_item_set_proxy_menu_item()` or "create-menu-proxy", will be available in an overflow menu, which can be opened by an added arrow button. If `FALSE`, `toolbar` will request enough size to fit all of its child items without any overflow.

Parameters

toolbar	a GtkToolbar
show_arrow	Whether to show an overflow menu

Since: 2.4

---

gtk_toolbar_unset_icon_size ()

```
void
gtk_toolbar_unset_icon_size (GtkToolbar *toolbar);
```

Unsets toolbar icon size set with `gtk_toolbar_set_icon_size()`, so that user preferences will be used to determine the icon size.

Parameters

toolbar	a GtkToolbar

---

gtk_toolbar_get_show_arrow ()

```
gboolean
gtk_toolbar_get_show_arrow (GtkToolbar *toolbar);
```

Returns whether the toolbar has an overflow menu. See `gtk_toolbar_set_show_arrow()`.

Parameters

toolbar	a GtkToolbar

Returns

`TRUE` if the toolbar has an overflow menu.

Since: 2.4

---

gtk_toolbar_get_style ()

```
GtkToolbarStyle
gtk_toolbar_get_style (GtkToolbar *toolbar);
```

Retrieves whether the toolbar has text, icons, or both . See `gtk_toolbar_set_style()`.

Parameters

toolbar	a GtkToolbar

Returns

the current style of `toolbar`

---

gtk_toolbar_get_icon_size ()

```
GtkIconSize
gtk_toolbar_get_icon_size (GtkToolbar *toolbar);
```

Retrieves the icon size for the toolbar. See `gtk_toolbar_set_icon_size()`.

Parameters

toolbar	a GtkToolbar

Returns

the current icon size for the icons on the toolbar.

---

gtk_toolbar_get_relief_style ()

```
GtkReliefStyle
gtk_toolbar_get_relief_style (GtkToolbar *toolbar);
```

Returns the relief style of buttons on `toolbar` . See `gtk_button_set_relief()`.

Parameters

toolbar	a GtkToolbar

Returns

The relief style of buttons on `toolbar` .

Since: 2.4

---

gtk_toolbar_set_style ()

```
void
gtk_toolbar_set_style (GtkToolbar *toolbar,
                       GtkToolbarStyle style);
```

Alters the view of `toolbar` to display either icons only, text only, or both.

Parameters

toolbar	a GtkToolbar.
style	the new style for `toolbar` .

---

gtk_toolbar_set_icon_size ()

```
void
gtk_toolbar_set_icon_size (GtkToolbar *toolbar,
                           GtkIconSize icon_size);
```

This function sets the size of stock icons in the toolbar. You can call it both before you add the icons and after they've been added. The size you set will override user preferences for the default icon size.

This should only be used for special-purpose toolbars, normal application toolbars should respect the user preferences for the size of icons.

Parameters

toolbar	A GtkToolbar
icon_size	The GtkIconSize that stock icons in the toolbar shall have.

---

gtk_toolbar_unset_style ()

```
void
gtk_toolbar_unset_style (GtkToolbar *toolbar);
```

Unsets a toolbar style set with `gtk_toolbar_set_style()`, so that user preferences will be used to determine the toolbar style.

Parameters

toolbar	a GtkToolbar

Types and Values

struct GtkToolbar

```
struct GtkToolbar;
```

---

enum GtkToolbarSpaceStyle

`GtkToolbarSpaceStyle` has been deprecated since version 3.20 and should not be used in newly-written code.

Whether spacers are vertical lines or just blank.

Members

GTK_TOOLBAR_SPACE_EMPTY	Use blank spacers.
GTK_TOOLBAR_SPACE_LINE	Use vertical lines for spacers.

Property Details

The `"icon-size"` property

`"icon-size"` `GtkIconSize`

The size of the icons in a toolbar is normally determined by the toolbar-icon-size setting. When this property is set, it overrides the setting.

This should only be used for special-purpose toolbars, normal application toolbars should respect the user preferences for the size of icons.

Flags: Read / Write

Default value: GTK_ICON_SIZE_LARGE_TOOLBAR

Since: 2.10

The `"icon-size-set"` property

`"icon-size-set"` `gboolean`

Is `TRUE` if the icon-size property has been set.

Flags: Read / Write

Default value: FALSE

Since: 2.10

The “show-arrow” property

“show-arrow” gboolean

If an arrow should be shown if the toolbar doesn't fit.

Flags: Read / Write

Default value: TRUE

The “toolbar-style” property

“toolbar-style” GtkToolbarStyle

How to draw the toolbar.

Flags: Read / Write

Default value: GTK_TOOLBAR_BOTH_HORIZ

Child Property Details

The “expand” child property

“expand” gboolean

Whether the item should receive extra space when the toolbar grows.

Flags: Read / Write

Default value: FALSE

The “homogeneous” child property

“homogeneous” gboolean

Whether the item should be the same size as other homogeneous items.

Flags: Read / Write

Default value: FALSE

Style Property Details

The “button-relief” style property

“button-relief” GtkReliefStyle

Type of bevel around toolbar buttons.

Flags: Read

Default value: GTK_RELIEF_NONE

The “internal-padding” style property

“internal-padding” gint

Amount of border space between the toolbar shadow and the buttons.

GtkToolbar:internal-padding has been deprecated since version 3.6 and should not be used in newly-written code.

Use the standard padding CSS property (through objects like GtkStyleContext and GtkCssProvider); the value of this style property is ignored.

Flags: Read

Allowed values: >= 0

Default value: 0

The “max-child-expand” style property

“max-child-expand” gint

Maximum amount of space an expandable item will be given.

Flags: Read

Allowed values: >= 0

Default value: 2147483647

The “shadow-type” style property

“shadow-type” GtkShadowType

Style of bevel around the toolbar.

GtkToolbar:shadow-type has been deprecated since version 3.6 and should not be used in newly-written code.

Use the standard border CSS property (through objects like GtkStyleContext and GtkCssProvider); the value of this style property is ignored.

Flags: Read

Default value: GTK_SHADOW_OUT

The “space-size” style property

“space-size” gint

Size of toolbar spacers.

GtkToolbar:space-size has been deprecated since version 3.20 and should not be used in newly-written code.

Use the standard margin/padding CSS properties on the separator elements; the value of this style property is ignored.

Flags: Read

Allowed values: >= 0

Default value: 12

The “space-style” style property

“space-style” GtkToolbarSpaceStyle

Style of toolbar spacers.

GtkToolbar:space-style has been deprecated since version 3.20 and should not be used in newly-written code.

Use CSS properties on the separator elements to style toolbar spacers; the value of this style property is ignored.

Flags: Read

Default value: GTK_TOOLBAR_SPACE_LINE

Signal Details

The “focus-home-or-end” signal

```
gboolean
user_function (GtkToolbar *toolbar,
               gboolean    focus_home,
               gpointer    user_data)
```

A keybinding signal used internally by GTK+. This signal can't be used in application code

Parameters

toolbar	the GtkToolbar which emitted the signal
focus_home	TRUE if the first item should be focused
user_data	user data set when the signal handler was connected.

Returns

TRUE if the signal was handled, FALSE if not

Flags: Action

The "orientation-changed" signal

```
void
user_function (GtkToolbar    *toolbar,
               GtkOrientation orientation,
               gpointer       user_data)
```

Emitted when the orientation of the toolbar changes.

Parameters

toolbar	the object which emitted the signal
orientation	the new GtkOrientation of the toolbar
user_data	user data set when the signal handler was connected.

Flags: Run First

The "popup-context-menu" signal

```
gboolean
user_function (GtkToolbar *toolbar,
               gint        x,
               gint        y,
               gint        button,
               gpointer    user_data)
```

Emitted when the user right-clicks the toolbar or uses the keybinding to display a popup menu.

Application developers should handle this signal if they want to display a context menu on the toolbar. The context-menu should appear at the coordinates given by x and y . The mouse button number is given by the button parameter. If the menu was popped up using the keybaord, button is -1.

Parameters

toolbar	the GtkToolbar which emitted the signal
x	the x coordinate of the point where the menu should appear
y	the y coordinate of the point where the menu should appear
button	the mouse button the user pressed, or -1
user_data	user data set when the signal handler was connected.

Returns

return `TRUE` if the signal was handled, `FALSE` if not

Flags: Run Last

The `"style-changed"` signal

```
void
user_function (GtkToolbar     *toolbar,
               GtkToolbarStyle style,
               gpointer        user_data)
```

Emitted when the style of the toolbar changes.

Parameters

toolbar	The GtkToolbar which emitted the signal
style	the new GtkToolbarStyle of the toolbar
user_data	user data set when the signal handler was connected.

Flags: Run First

See Also

GtkToolItem

GtkToolItem

GtkToolItem — The base class of widgets that can be added to GtkToolShell

Functions

GtkToolItem *	gtk_tool_item_new ()
void	gtk_tool_item_set_homogeneous ()
gboolean	gtk_tool_item_get_homogeneous ()
void	gtk_tool_item_set_expand ()
gboolean	gtk_tool_item_get_expand ()
void	gtk_tool_item_set_tooltip_text ()
void	gtk_tool_item_set_tooltip_markup ()
void	gtk_tool_item_set_use_drag_window ()
gboolean	gtk_tool_item_get_use_drag_window ()
void	gtk_tool_item_set_visible_horizontal ()

gboolean	gtk_tool_item_get_visible_horizontal ()
void	gtk_tool_item_set_visible_vertical ()
gboolean	gtk_tool_item_get_visible_vertical ()
void	gtk_tool_item_set_is_important ()
gboolean	gtk_tool_item_get_is_important ()
PangoEllipsizeMode	gtk_tool_item_get_ellipsize_mode ()
GtkIconSize	gtk_tool_item_get_icon_size ()
GtkOrientation	gtk_tool_item_get_orientation ()
GtkToolbarStyle	gtk_tool_item_get_toolbar_style ()
GtkReliefStyle	gtk_tool_item_get_relief_style ()
gfloat	gtk_tool_item_get_text_alignment ()
GtkOrientation	gtk_tool_item_get_text_orientation ()
GtkWidget *	gtk_tool_item_retrieve_proxy_menu_item ()
GtkWidget *	gtk_tool_item_get_proxy_menu_item ()
void	gtk_tool_item_set_proxy_menu_item ()
void	gtk_tool_item_rebuild_menu ()
void	gtk_tool_item_toolbar_reconfigured ()
GtkSizeGroup *	gtk_tool_item_get_text_size_group ()

Properties

gboolean	is-important	Read / Write
gboolean	visible-horizontal	Read / Write
gboolean	visible-vertical	Read / Write

Signals

gboolean	create-menu-proxy	Run Last
void	toolbar-reconfigured	Run Last

Types and Values

struct	GtkToolItem
struct	GtkToolItemClass

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkContainer
            └── GtkBin
                └── GtkToolItem
                    ├── GtkToolButton
                    └── GtkSeparatorToolItem
```

Implemented Interfaces

GtkToolItem implements AtkImplementorIface, GtkBuildable and GtkActivatable.

Includes

```
#include <gtk/gtk.h>
```

Description

GtkToolItems are widgets that can appear on a toolbar. To create a toolbar item that contain something else than a button, use `gtk_tool_item_new()`. Use `gtk_container_add()` to add a child widget to the tool item.

For toolbar items that contain buttons, see the GtkToolButton, GtkToggleToolButton and GtkRadioToolButton classes.

See the GtkToolbar class for a description of the toolbar widget, and GtkToolShell for a description of the tool shell interface.

Functions

gtk_tool_item_new ()

```
GtkToolItem *
gtk_tool_item_new (void);
```

Creates a new GtkToolItem

Returns

the new GtkToolItem

Since: 2.4

---

gtk_tool_item_set_homogeneous ()

```
void
gtk_tool_item_set_homogeneous (GtkToolItem *tool_item,
                               gboolean homogeneous);
```

Sets whether `tool_item` is to be allocated the same size as other homogeneous items. The effect is that all homogeneous items will have the same width as the widest of the items.

Parameters

tool_item	a GtkToolItem
homogeneous	whether `tool_item` is the same size as other homogeneous items

Since: 2.4

---

gtk_tool_item_get_homogeneous ()

```
gboolean
gtk_tool_item_get_homogeneous (GtkToolItem *tool_item);
```

Returns whether `tool_item` is the same size as other homogeneous items. See `gtk_tool_item_set_homogeneous()`.

Parameters

tool_item	a GtkToolItem

Returns

`TRUE` if the item is the same size as other homogeneous items.

Since: 2.4

---

gtk_tool_item_set_expand ()

```
void
gtk_tool_item_set_expand (GtkToolItem *tool_item,
                          gboolean expand);
```

Sets whether `tool_item` is allocated extra space when there is more room on the toolbar then needed for the items. The effect is that the item gets bigger when the toolbar gets bigger and smaller when the toolbar gets smaller.

Parameters

tool_item	a GtkToolItem
expand	Whether `tool_item` is allocated extra space

Since: 2.4

---

gtk_tool_item_get_expand ()

```
gboolean
gtk_tool_item_get_expand (GtkToolItem *tool_item);
```

Returns whether `tool_item` is allocated extra space. See `gtk_tool_item_set_expand()`.

Parameters

tool_item	a GtkToolItem

Returns

`TRUE` if `tool_item` is allocated extra space.

Since: 2.4

---

gtk_tool_item_set_tooltip_text ()

```
void
gtk_tool_item_set_tooltip_text (GtkToolItem *tool_item,
                                const gchar *text);
```

Sets the text to be displayed as tooltip on the item. See `gtk_widget_set_tooltip_text()`.

Parameters

tool_item	a GtkToolItem
text	text to be used as tooltip for `tool_item`

Since: 2.12

---

gtk_tool_item_set_tooltip_markup ()

```
void
gtk_tool_item_set_tooltip_markup (GtkToolItem *tool_item,
                                  const gchar *markup);
```

Sets the markup text to be displayed as tooltip on the item. See `gtk_widget_set_tooltip_markup()`.

Parameters

tool_item	a GtkToolItem
markup	markup text to be used as tooltip for `tool_item`

Since: 2.12

---

gtk_tool_item_set_use_drag_window ()

```
void
gtk_tool_item_set_use_drag_window (GtkToolItem *tool_item,
                                   gboolean use_drag_window);
```

Sets whether `tool_item` has a drag window. When `TRUE` the toolitem can be used as a drag source through `gtk_drag_source_set()`. When `tool_item` has a drag window it will intercept all events, even those that would otherwise be sent to a child of `tool_item` .

Parameters

tool_item	a GtkToolItem
use_drag_window	Whether `tool_item` has a drag window.

Since: 2.4

---

gtk_tool_item_get_use_drag_window ()

```
gboolean
gtk_tool_item_get_use_drag_window (GtkToolItem *tool_item);
```

Returns whether `tool_item` has a drag window. See `gtk_tool_item_set_use_drag_window()`.

Parameters

tool_item	a GtkToolItem

Returns

`TRUE` if `tool_item` uses a drag window.

Since: 2.4

---

gtk_tool_item_set_visible_horizontal ()

```
void
gtk_tool_item_set_visible_horizontal (GtkToolItem *tool_item,
                                      gboolean visible_horizontal);
```

Sets whether `tool_item` is visible when the toolbar is docked horizontally.

Parameters

tool_item	a GtkToolItem
visible_horizontal	Whether `tool_item` is visible when in horizontal mode

Since: 2.4

---

gtk_tool_item_get_visible_horizontal ()

```
gboolean
gtk_tool_item_get_visible_horizontal (GtkToolItem *tool_item);
```

Returns whether the `tool_item` is visible on toolbars that are docked horizontally.

Parameters

tool_item	a GtkToolItem

Returns

`TRUE` if `tool_item` is visible on toolbars that are docked horizontally.

Since: 2.4

---

gtk_tool_item_set_visible_vertical ()

```
void
gtk_tool_item_set_visible_vertical (GtkToolItem *tool_item,
                                    gboolean visible_vertical);
```

Sets whether `tool_item` is visible when the toolbar is docked vertically. Some tool items, such as text entries, are too wide to be useful on a vertically docked toolbar. If `visible_vertical` is `FALSE` `tool_item` will not appear on toolbars that are docked vertically.

Parameters

tool_item	a GtkToolItem
visible_vertical	whether `tool_item` is visible when the toolbar is in vertical mode

Since: 2.4

---

gtk_tool_item_get_visible_vertical ()

```
gboolean
gtk_tool_item_get_visible_vertical (GtkToolItem *tool_item);
```

Returns whether `tool_item` is visible when the toolbar is docked vertically. See `gtk_tool_item_set_visible_vertical()`.

Parameters

tool_item	a GtkToolItem

Returns

Whether `tool_item` is visible when the toolbar is docked vertically

Since: 2.4

---

gtk_tool_item_set_is_important ()

```
void
gtk_tool_item_set_is_important (GtkToolItem *tool_item,
                                gboolean is_important);
```

Sets whether `tool_item` should be considered important. The GtkToolButton class uses this property to determine whether to show or hide its label when the toolbar style is `GTK_TOOLBAR_BOTH_HORIZ`. The result is

that only tool buttons with the “is_important” property set have labels, an effect known as “priority text”

Parameters

tool_item	a GtkToolItem
is_important	whether the tool item should be considered important

Since: 2.4

---

gtk_tool_item_get_is_important ()

```
gboolean
gtk_tool_item_get_is_important (GtkToolItem *tool_item);
```

Returns whether `tool_item` is considered important. See `gtk_tool_item_set_is_important()`

Parameters

tool_item	a GtkToolItem

Returns

`TRUE` if `tool_item` is considered important.

Since: 2.4

---

gtk_tool_item_get_ellipsize_mode ()

```
PangoEllipsizeMode
gtk_tool_item_get_ellipsize_mode (GtkToolItem *tool_item);
```

Returns the ellipsize mode used for `tool_item` . Custom subclasses of GtkToolItem should call this function to find out how text should be ellipsized.

Parameters

tool_item	a GtkToolItem

Returns

a PangoEllipsizeMode indicating how text in `tool_item` should be ellipsized.

Since: 2.20

---

gtk_tool_item_get_icon_size ()

```
GtkIconSize
gtk_tool_item_get_icon_size (GtkToolItem *tool_item);
```

Returns the icon size used for `tool_item` . Custom subclasses of GtkToolItem should call this function to find out what size icons they should use.

Parameters

tool_item	a GtkToolItem

Returns

a GtkIconSize indicating the icon size used for `tool_item` .

[type int]

Since: 2.4

---

gtk_tool_item_get_orientation ()

```
GtkOrientation
gtk_tool_item_get_orientation (GtkToolItem *tool_item);
```

Returns the orientation used for `tool_item` . Custom subclasses of GtkToolItem should call this function to find out what size icons they should use.

Parameters

tool_item	a GtkToolItem

Returns

a GtkOrientation indicating the orientation used for `tool_item`

Since: 2.4

---

gtk_tool_item_get_toolbar_style ()

```
GtkToolbarStyle
gtk_tool_item_get_toolbar_style (GtkToolItem *tool_item);
```

Returns the toolbar style used for `tool_item` . Custom subclasses of GtkToolItem should call this function in the handler of the GtkToolItem::toolbar_reconfigured signal to find out in what style the toolbar is displayed and change themselves accordingly

Possibilities are:

- `GTK_TOOLBAR_BOTH`, meaning the tool item should show both an icon and a label, stacked vertically
- `GTK_TOOLBAR_ICONS`, meaning the toolbar shows only icons

- GTK_TOOLBAR_TEXT, meaning the tool item should only show text
- GTK_TOOLBAR_BOTH_HORIZ, meaning the tool item should show both an icon and a label, arranged horizontally

Parameters

tool_item	a GtkToolItem

Returns

A GtkToolbarStyle indicating the toolbar style used for `tool_item` .

Since: 2.4

---

gtk_tool_item_get_relief_style ()

```
GtkReliefStyle
gtk_tool_item_get_relief_style (GtkToolItem *tool_item);
```

Returns the relief style of `tool_item` . See `gtk_button_set_relief()`. Custom subclasses of GtkToolItem should call this function in the handler of the "toolbar_reconfigured" signal to find out the relief style of buttons.

Parameters

tool_item	a GtkToolItem

Returns

a GtkReliefStyle indicating the relief style used for `tool_item` .

Since: 2.4

---

gtk_tool_item_get_text_alignment ()

```
gfloat
gtk_tool_item_get_text_alignment (GtkToolItem *tool_item);
```

Returns the text alignment used for `tool_item` . Custom subclasses of GtkToolItem should call this function to find out how text should be aligned.

Parameters

tool_item	a GtkToolItem:

Returns

a gfloat indicating the horizontal text alignment used for `tool_item`

Since: 2.20

---

gtk_tool_item_get_text_orientation ()

```
GtkOrientation
gtk_tool_item_get_text_orientation (GtkToolItem *tool_item);
```

Returns the text orientation used for `tool_item` . Custom subclasses of GtkToolItem should call this function to find out how text should be orientated.

Parameters

tool_item	a GtkToolItem

Returns

a GtkOrientation indicating the text orientation used for `tool_item`

Since: 2.20

---

gtk_tool_item_retrieve_proxy_menu_item ()

```
GtkWidget *
gtk_tool_item_retrieve_proxy_menu_item
                                (GtkToolItem *tool_item);
```

Returns the GtkMenuItem that was last set by `gtk_tool_item_set_proxy_menu_item()`, ie. the GtkMenuItem that is going to appear in the overflow menu.

Parameters

tool_item	a GtkToolItem

Returns

The GtkMenuItem that is going to appear in the overflow menu for `tool_item` .

[transfer none]

Since: 2.4

---

gtk_tool_item_get_proxy_menu_item ()

```
GtkWidget *
```

```
gtk_tool_item_get_proxy_menu_item (GtkToolItem *tool_item,
                                   const gchar *menu_item_id);
```

If `menu_item_id` matches the string passed to `gtk_tool_item_set_proxy_menu_item()` return the corresponding GtkMenuItem.

Custom subclasses of GtkToolItem should use this function to update their menu item when the GtkToolItem changes. That the `menu_item_ids` must match ensures that a GtkToolItem will not inadvertently change a menu item that they did not create.

Parameters

tool_item	a GtkToolItem	
menu_item_id	a string used to identify the menu item	

Returns

The GtkMenuItem passed to `gtk_tool_item_set_proxy_menu_item()`, if the `menu_item_ids` match.

[transfer none][nullable]

Since: 2.4

---

gtk_tool_item_set_proxy_menu_item ()

```
void
gtk_tool_item_set_proxy_menu_item (GtkToolItem *tool_item,
                                   const gchar *menu_item_id,
                                   GtkWidget *menu_item);
```

Sets the GtkMenuItem used in the toolbar overflow menu. The `menu_item_id` is used to identify the caller of this function and should also be used with `gtk_tool_item_get_proxy_menu_item()`.

See also "create-menu-proxy".

Parameters

tool_item	a GtkToolItem	
menu_item_id	a string used to identify `menu_item`	
menu_item	a GtkMenuItem to use in the overflow menu, or `NULL`.	[nullable]

Since: 2.4

---

gtk_tool_item_rebuild_menu ()

```
void
gtk_tool_item_rebuild_menu (GtkToolItem *tool_item);
```

Calling this function signals to the toolbar that the overflow menu item for `tool_item` has changed. If the overflow menu is visible when this function it called, the menu will be rebuilt.

The function must be called when the tool item changes what it will do in response to the “create-menu-proxy” signal.

Parameters

tool_item	a GtkToolItem

Since: 2.6

---

gtk_tool_item_toolbar_reconfigured ()

```
void
gtk_tool_item_toolbar_reconfigured (GtkToolItem *tool_item);
```

Emits the signal “toolbar_reconfigured” on `tool_item`. GtkToolbar and other GtkToolShell implementations use this function to notify children, when some aspect of their configuration changes.

Parameters

tool_item	a GtkToolItem

Since: 2.14

---

gtk_tool_item_get_text_size_group ()

```
GtkSizeGroup *
gtk_tool_item_get_text_size_group (GtkToolItem *tool_item);
```

Returns the size group used for labels in `tool_item`. Custom subclasses of GtkToolItem should call this function and use the size group for labels.

Parameters

tool_item	a GtkToolItem

Returns

a GtkSizeGroup.

[transfer none]

Since: 2.20

Types and Values

struct GtkToolItem

```
struct GtkToolItem;
```

The GtkToolItem struct contains only private data. It should only be accessed through the functions described below.

---

struct GtkToolItemClass

```
struct GtkToolItemClass {
  GtkBinClass parent_class;

  /* signals */
  gboolean   (* create_menu_proxy)    (GtkToolItem *tool_item);
  void       (* toolbar_reconfigured) (GtkToolItem *tool_item);
};
```

Members

`create_menu_proxy` ()	Signal emitted when the toolbar needs information from tool_item about whether the item should appear in the toolbar overflow menu.
`toolbar_reconfigured` ()	Signal emitted when some property of the toolbar that the item is a child of changes.

Property Details

The `"is-important"` property

`"is-important"` `gboolean`

Whether the toolbar item is considered important. When TRUE, toolbar buttons show text in GTK_TOOLBAR_BOTH_HORIZ mode.

Flags: Read / Write

Default value: FALSE

The `"visible-horizontal"` property

`"visible-horizontal"` `gboolean`

Whether the toolbar item is visible when the toolbar is in a horizontal orientation.

Flags: Read / Write

Default value: TRUE

The “visible-vertical” property

```
  “visible-vertical”         gboolean
```

Whether the toolbar item is visible when the toolbar is in a vertical orientation.

Flags: Read / Write

Default value: TRUE

Signal Details

The “create-menu-proxy” signal

```
gboolean
user_function (GtkToolItem *tool_item,
               gpointer     user_data)
```

This signal is emitted when the toolbar needs information from `tool_item` about whether the item should appear in the toolbar overflow menu. In response the tool item should either

- call `gtk_tool_item_set_proxy_menu_item()` with a `NULL` pointer and return `TRUE` to indicate that the item should not appear in the overflow menu
- call `gtk_tool_item_set_proxy_menu_item()` with a new menu item and return `TRUE`, or
- return `FALSE` to indicate that the signal was not handled by the item. This means that the item will not appear in the overflow menu unless a later handler installs a menu item.

The toolbar may cache the result of this signal. When the tool item changes how it will respond to this signal it must call `gtk_tool_item_rebuild_menu()` to invalidate the cache and ensure that the toolbar rebuilds its overflow menu.

Parameters

tool_item	the object the signal was emitted on
user_data	user data set when the signal handler was connected.

Returns

`TRUE` if the signal was handled, `FALSE` if not

Flags: Run Last

The “toolbar-reconfigured” signal

```
void
user_function (GtkToolItem *tool_item,
               gpointer     user_data)
```

This signal is emitted when some property of the toolbar that the item is a child of changes. For custom

subclasses of GtkToolItem, the default handler of this signal use the functions

- gtk_tool_shell_get_orientation()
- gtk_tool_shell_get_style()
- gtk_tool_shell_get_icon_size()
- gtk_tool_shell_get_relief_style() to find out what the toolbar should look like and change themselves accordingly.

Parameters

tool_item	the object the signal was emitted on
user_data	user data set when the signal handler was connected.

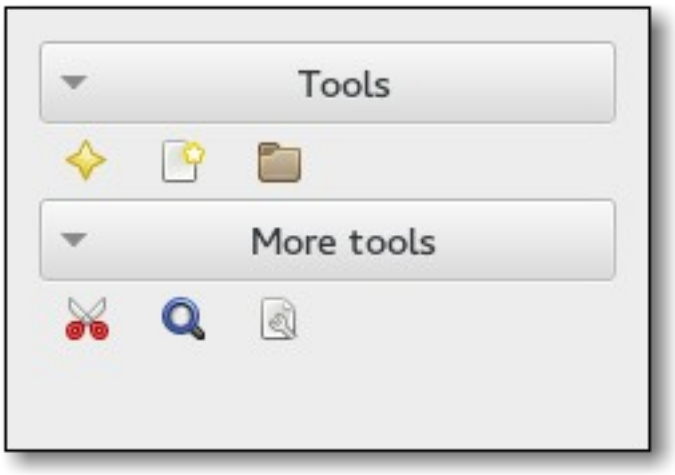
Flags: Run Last

See Also

GtkToolbar, GtkToolButton, GtkSeparatorToolItem

GtkToolPalette

GtkToolPalette — A tool palette with categories



Functions

GtkWidget *	gtk_tool_palette_new ()
gboolean	gtk_tool_palette_get_exclusive ()
void	gtk_tool_palette_set_exclusive ()
gboolean	gtk_tool_palette_get_expand ()
void	gtk_tool_palette_set_expand ()
gint	gtk_tool_palette_get_group_position ()
void	gtk_tool_palette_set_group_position ()
GtkIconSize	gtk_tool_palette_get_icon_size ()
void	gtk_tool_palette_set_icon_size ()
void	gtk_tool_palette_unset_icon_size ()
GtkToolbarStyle	gtk_tool_palette_get_style ()
void	gtk_tool_palette_set_style ()

void	gtk_tool_palette_unset_style ()
void	gtk_tool_palette_add_drag_dest ()
GtkWidget *	gtk_tool_palette_get_drag_item ()
const GtkTargetEntry *	gtk_tool_palette_get_drag_target_group ()
const GtkTargetEntry *	gtk_tool_palette_get_drag_target_item ()
GtkToolItemGroup *	gtk_tool_palette_get_drop_group ()
GtkToolItem *	gtk_tool_palette_get_drop_item ()
void	gtk_tool_palette_set_drag_source ()
GtkAdjustment *	gtk_tool_palette_get_hadjustment ()
GtkAdjustment *	gtk_tool_palette_get_vadjustment ()

Properties

GtkIconSize	icon-size	Read / Write
gboolean	icon-size-set	Read / Write
GtkToolbarStyle	toolbar-style	Read / Write

Child Properties

gboolean	exclusive	Read / Write
gboolean	expand	Read / Write

Types and Values

struct	GtkToolPalette
struct	GtkToolPaletteClass
enum	GtkToolPaletteDragTargets

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkContainer
            └── GtkToolPalette
```

Implemented Interfaces

GtkToolPalette implements AtkImplementorIface, GtkBuildable, GtkOrientable and GtkScrollable.

Includes

```
#include <gtk/gtk.h>
```

Description

A GtkToolPalette allows you to add GtkToolItems to a palette-like container with different categories and drag and drop support.

A GtkToolPalette is created with a call to `gtk_tool_palette_new()`.

GtkToolItems cannot be added directly to a GtkToolPalette - instead they are added to a GtkToolItemGroup which can than be added to a GtkToolPalette. To add a GtkToolItemGroup to a GtkToolPalette, use `gtk_container_add()`.

```
GtkWidget *palette, *group;
GtkToolItem *item;

palette = gtk_tool_palette_new ();
group = gtk_tool_item_group_new (_("Test
Category"));
gtk_container_add (GTK_CONTAINER (palette),
group);

item = gtk_tool_button_new (NULL,
_("_Open"));
gtk_tool_button_set_icon_name
(GTK_TOOL_BUTTON (item), "document-open");
gtk_tool_item_group_insert
(GTK_TOOL_ITEM_GROUP (group), item, -1);
```

The easiest way to use drag and drop with GtkToolPalette is to call `gtk_tool_palette_add_drag_dest()` with the desired drag source `palette` and the desired drag target `widget` . Then `gtk_tool_palette_get_drag_item()` can be used to get the dragged item in the "drag-data-received" signal handler of the drag target.

```
static void
passive_canvas_drag_data_received (GtkWidget
*widget,

GdkDragContext   *context,
                                        gint
x,
                                        gint
y,

GtkSelectionData *selection,
                                        guint
info,
                                        guint
time,
                                        gpointer
data)
{
  GtkWidget *palette;
  GtkWidget *item;

  // Get the dragged item
  palette = gtk_widget_get_ancestor
(gtk_drag_get_source_widget (context),

GTK_TYPE_TOOL_PALETTE);
  if (palette != NULL)
    item = gtk_tool_palette_get_drag_item
(GTK_TOOL_PALETTE (palette),

selection);

  // Do something with item
}

GtkWidget *target, palette;
```

```
palette = gtk_tool_palette_new ();
target = gtk_drawing_area_new ();

g_signal_connect (G_OBJECT (target), "drag-
data-received",
                  G_CALLBACK
(passive_canvas_drag_data_received), NULL);
gtk_tool_palette_add_drag_dest
(GTK_TOOL_PALETTE (palette), target,

GTK_DEST_DEFAULT_ALL,

GTK_TOOL_PALETTE_DRAG_ITEMS,

GDK_ACTION_COPY);
```

CSS nodes

GtkToolPalette has a single CSS node named toolpalette.

Functions

gtk_tool_palette_new ()

```
GtkWidget *
gtk_tool_palette_new (void);
```

Creates a new tool palette.

Returns

a new GtkToolPalette

Since: 2.20

---

gtk_tool_palette_get_exclusive ()

```
gboolean
gtk_tool_palette_get_exclusive (GtkToolPalette *palette,
                                GtkToolItemGroup *group);
```

Gets whether `group` is exclusive or not. See `gtk_tool_palette_set_exclusive()`.

Parameters

palette	a GtkToolPalette
group	a GtkToolItemGroup which is a child of palette

Returns

`TRUE` if `group` is exclusive

Since: 2.20

---

gtk_tool_palette_set_exclusive ()

```
void
gtk_tool_palette_set_exclusive (GtkToolPalette *palette,
                                GtkToolItemGroup *group,
                                gboolean exclusive);
```

Sets whether the group should be exclusive or not. If an exclusive group is expanded all other groups are collapsed.

Parameters

palette	a GtkToolPalette
group	a GtkToolItemGroup which is a child of palette
exclusive	whether the group should be exclusive or not

Since: 2.20

---

gtk_tool_palette_get_expand ()

```
gboolean
gtk_tool_palette_get_expand (GtkToolPalette *palette,
                             GtkToolItemGroup *group);
```

Gets whether group should be given extra space. See `gtk_tool_palette_set_expand()`.

Parameters

palette	a GtkToolPalette
group	a GtkToolItemGroup which is a child of palette

Returns

`TRUE` if group should be given extra space, `FALSE` otherwise

Since: 2.20

---

gtk_tool_palette_set_expand ()

```
void
```

```
gtk_tool_palette_set_expand (GtkToolPalette *palette,
                             GtkToolItemGroup *group,
                             gboolean expand);
```

Sets whether the group should be given extra space.

Parameters

palette	a GtkToolPalette
group	a GtkToolItemGroup which is a child of palette
expand	whether the group should be given extra space

Since: 2.20

---

gtk_tool_palette_get_group_position ()

```
gint
gtk_tool_palette_get_group_position (GtkToolPalette *palette,
                                     GtkToolItemGroup *group);
```

Gets the position of `group` in `palette` as index. See `gtk_tool_palette_set_group_position()`.

Parameters

palette	a GtkToolPalette
group	a GtkToolItemGroup

Returns

the index of group or -1 if `group` is not a child of `palette`

Since: 2.20

---

gtk_tool_palette_set_group_position ()

```
void
gtk_tool_palette_set_group_position (GtkToolPalette *palette,
                                     GtkToolItemGroup *group,
                                     gint position);
```

Sets the position of the group as an index of the tool palette. If position is 0 the group will become the first child, if position is -1 it will become the last child.

Parameters

palette	a GtkToolPalette
group	a GtkToolItemGroup which is a child of palette
position	a new index for group

Since: 2.20

---

gtk_tool_palette_get_icon_size ()

```
GtkIconSize
gtk_tool_palette_get_icon_size (GtkToolPalette *palette);
```

Gets the size of icons in the tool palette. See `gtk_tool_palette_set_icon_size()`.

Parameters

palette	a GtkToolPalette	

Returns

the GtkIconSize of icons in the tool palette.

[type int]

Since: 2.20

---

gtk_tool_palette_set_icon_size ()

```
void
gtk_tool_palette_set_icon_size (GtkToolPalette *palette,
                                GtkIconSize icon_size);
```

Sets the size of icons in the tool palette.

Parameters

palette	a GtkToolPalette	
icon_size	the GtkIconSize that icons in the tool palette shall have.	[type int]

Since: 2.20

---

gtk_tool_palette_unset_icon_size ()

```
void
gtk_tool_palette_unset_icon_size (GtkToolPalette *palette);
```

Unsets the tool palette icon size set with `gtk_tool_palette_set_icon_size()`, so that user preferences will be used to determine the icon size.

Parameters

palette	a GtkToolPalette	

Since: 2.20

---

gtk_tool_palette_get_style ()

```
GtkToolbarStyle
gtk_tool_palette_get_style (GtkToolPalette *palette);
```

Gets the style (icons, text or both) of items in the tool palette.

Parameters

palette	a GtkToolPalette

Returns

the GtkToolbarStyle of items in the tool palette.

Since: 2.20

---

gtk_tool_palette_set_style ()

```
void
gtk_tool_palette_set_style (GtkToolPalette *palette,
                            GtkToolbarStyle style);
```

Sets the style (text, icons or both) of items in the tool palette.

Parameters

palette	a GtkToolPalette
style	the GtkToolbarStyle that items in the tool palette shall have

Since: 2.20

---

gtk_tool_palette_unset_style ()

```
void
gtk_tool_palette_unset_style (GtkToolPalette *palette);
```

Unsets a toolbar style set with `gtk_tool_palette_set_style()`, so that user preferences will be used to determine the toolbar style.

Parameters

palette	a GtkToolPalette

Since: 2.20

---

gtk_tool_palette_add_drag_dest ()

```
void
gtk_tool_palette_add_drag_dest (GtkToolPalette *palette,
                                GtkWidget *widget,
                                GtkDestDefaults flags,
                                GtkToolPaletteDragTargets targets,
                                GdkDragAction actions);
```

Sets `palette` as drag source (see `gtk_tool_palette_set_drag_source()`) and sets `widget` as a drag destination for drags from `palette` . See `gtk_drag_dest_set()`.

Parameters

palette	a GtkToolPalette
widget	a GtkWidget which should be a drag destination for `palette`
flags	the flags that specify what actions GTK+ should take for drops on that widget
targets	the GtkToolPaletteDragTargets which the widget should support
actions	the GdkDragActions which the widget should suppport

Since: 2.20

---

gtk_tool_palette_get_drag_item ()

```
GtkWidget *
gtk_tool_palette_get_drag_item (GtkToolPalette *palette,
                                const GtkSelectionData *selection);
```

Get the dragged item from the selection. This could be a GtkToolItem or a GtkToolItemGroup.

Parameters

palette	a GtkToolPalette
selection	a GtkSelectionData

Returns

the dragged item in selection.

[transfer none]

Since: 2.20

---

gtk_tool_palette_get_drag_target_group ()

```
const GtkTargetEntry *
gtk_tool_palette_get_drag_target_group
                                (void);
```

Get the target entry for a dragged GtkToolItemGroup.

Returns

the GtkTargetEntry for a dragged group.

[transfer none]

Since: 2.20

---

gtk_tool_palette_get_drag_target_item ()

```
const GtkTargetEntry *
gtk_tool_palette_get_drag_target_item (void);
```

Gets the target entry for a dragged GtkToolItem.

Returns

the GtkTargetEntry for a dragged item.

[transfer none]

Since: 2.20

---

gtk_tool_palette_get_drop_group ()

```
GtkToolItemGroup *
gtk_tool_palette_get_drop_group (GtkToolPalette *palette,
                                 gint x,
                                 gint y);
```

Gets the group at position (x, y).

Parameters

palette	a GtkToolPalette
x	the x position
y	the y position

Returns

the GtkToolItemGroup at position or `NULL` if there is no such group.

[nullable][transfer none]

Since: 2.20

---

gtk_tool_palette_get_drop_item ()

```
GtkToolItem *
gtk_tool_palette_get_drop_item (GtkToolPalette *palette,
                                gint x,
                                gint y);
```

Gets the item at position (x, y). See `gtk_tool_palette_get_drop_group()`.

Parameters

palette	a GtkToolPalette
x	the x position
y	the y position

Returns

the GtkToolItem at position or `NULL` if there is no such item.

[nullable][transfer none]

Since: 2.20

---

gtk_tool_palette_set_drag_source ()

```
void
gtk_tool_palette_set_drag_source (GtkToolPalette *palette,
                                  GtkToolPaletteDragTargets targets);
```

Sets the tool palette as a drag source. Enables all groups and items in the tool palette as drag sources on button 1 and button 3 press with copy and move actions. See `gtk_drag_source_set()`.

Parameters

palette	a GtkToolPalette
targets	the GtkToolPaletteDragTargets which the widget should support

Since: 2.20

---

gtk_tool_palette_get_hadjustment ()

```
GtkAdjustment *
gtk_tool_palette_get_hadjustment (GtkToolPalette *palette);
```

`gtk_tool_palette_get_hadjustment` has been deprecated since version 3.0 and should not be used in newly-written code.

Use gtk_scrollable_get_hadjustment()

Gets the horizontal adjustment of the tool palette.

Parameters

palette	a GtkToolPalette

Returns

the horizontal adjustment of `palette` .

[transfer none]

Since: 2.20

---

gtk_tool_palette_get_vadjustment ()

```
GtkAdjustment *
gtk_tool_palette_get_vadjustment (GtkToolPalette *palette);
```

`gtk_tool_palette_get_vadjustment` has been deprecated since version 3.0 and should not be used in newly-written code.

Use gtk_scrollable_get_vadjustment()

Gets the vertical adjustment of the tool palette.

Parameters

palette	a GtkToolPalette

Returns

the vertical adjustment of `palette` .

[transfer none]

Since: 2.20

Types and Values

struct GtkToolPalette

```
struct GtkToolPalette;
```

This should not be accessed directly. Use the accessor functions below.

---

struct GtkToolPaletteClass

```
struct GtkToolPaletteClass {
  GtkContainerClass parent_class;
};
```

Members

---

enum GtkToolPaletteDragTargets

Flags used to specify the supported drag targets.

Members

GTK_TOOL_PALETTE_DRAG_ITEMS	Support drag of items.
GTK_TOOL_PALETTE_DRAG_GROUPS	Support drag of groups.

Property Details

The `“icon-size”` property

`“icon-size”` `GtkIconSize`

The size of the icons in a tool palette. When this property is set, it overrides the default setting.

This should only be used for special-purpose tool palettes, normal application tool palettes should respect the user preferences for the size of icons.

Flags: Read / Write

Default value: GTK_ICON_SIZE_SMALL_TOOLBAR

Since: 2.20

The `“icon-size-set”` property

`“icon-size-set”` `gboolean`

Is `TRUE` if the “icon-size” property has been set.

Flags: Read / Write

Default value: FALSE

Since: 2.20

The “toolbar-style” property

“toolbar-style” GtkToolbarStyle

The style of items in the tool palette.

Flags: Read / Write

Default value: GTK_TOOLBAR_ICONS

Since: 2.20

Child Property Details

The “exclusive” child property

“exclusive” gboolean

Whether the item group should be the only one that is expanded at a given time.

Flags: Read / Write

Default value: FALSE

Since: 2.20

The “expand” child property

“expand” gboolean

Whether the item group should receive extra space when the palette grows. at a given time.

Flags: Read / Write

Default value: FALSE

Since: 2.20

GtkToolItemGroup

GtkToolItemGroup — A sub container used in a tool palette

Functions

gboolean	gtk_tool_item_group_get_collapsed ()
GtkToolItem *	gtk_tool_item_group_get_drop_item ()
PangoEllipsizeMode	gtk_tool_item_group_get_ellipsize ()
gint	gtk_tool_item_group_get_item_position ()
guint	gtk_tool_item_group_get_n_items ()
const gchar *	gtk_tool_item_group_get_label ()
GtkWidget *	gtk_tool_item_group_get_label_widget ()

GtkToolItem *	gtk_tool_item_group_get_nth_item ()
GtkReliefStyle	gtk_tool_item_group_get_header_relief ()
void	gtk_tool_item_group_insert ()
GtkWidget *	gtk_tool_item_group_new ()
void	gtk_tool_item_group_set_collapsed ()
void	gtk_tool_item_group_set_ellipsize ()
void	gtk_tool_item_group_set_item_position ()
void	gtk_tool_item_group_set_label ()
void	gtk_tool_item_group_set_label_widget ()
void	gtk_tool_item_group_set_header_relief ()

Properties

gboolean	collapsed	Read / Write
PangoEllipsizeMode	ellipsize	Read / Write
GtkReliefStyle	header-relief	Read / Write
gchar *	label	Read / Write
GtkWidget *	label-widget	Read / Write

Child Properties

gboolean	expand	Read / Write
gboolean	fill	Read / Write
gboolean	homogeneous	Read / Write
gboolean	new-row	Read / Write
gint	position	Read / Write

Style Properties

gint	expander-size	Read
gint	header-spacing	Read

Types and Values

struct	GtkToolItemGroup
struct	GtkToolItemGroupClass

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkContainer
            └── GtkToolItemGroup
```

Implemented Interfaces

GtkToolItemGroup implements AtkImplementorIface, GtkBuildable and GtkToolShell.

Includes

```
#include <gtk/gtk.h>
```

Description

A GtkToolItemGroup is used together with GtkToolPalette to add GtkToolItems to a palette like container with different categories and drag and drop support.

CSS nodes

GtkToolItemGroup has a single CSS node named toolitemgroup.

Functions

gtk_tool_item_group_get_collapsed ()

```
gboolean
gtk_tool_item_group_get_collapsed (GtkToolItemGroup *group);
```

Gets whether `group` is collapsed or expanded.

Parameters

group	a GtkToolItemGroup

Returns

`TRUE` if `group` is collapsed, `FALSE` if it is expanded

Since: 2.20

---

gtk_tool_item_group_get_drop_item ()

```
GtkToolItem *
gtk_tool_item_group_get_drop_item (GtkToolItemGroup *group,
                                   gint x,
                                   gint y);
```

Gets the tool item at position (x, y).

Parameters

group	a GtkToolItemGroup
x	the x position
y	the y position

Returns

the GtkToolItem at position (x, y).

[transfer none]

Since: 2.20

---

gtk_tool_item_group_get_ellipsize ()

```
PangoEllipsizeMode
gtk_tool_item_group_get_ellipsize (GtkToolItemGroup *group);
```

Gets the ellipsization mode of `group` .

Parameters

group	a GtkToolItemGroup

Returns

the PangoEllipsizeMode of `group`

Since: 2.20

---

gtk_tool_item_group_get_item_position ()

```
gint
gtk_tool_item_group_get_item_position (GtkToolItemGroup *group,
                                       GtkToolItem *item);
```

Gets the position of `item` in `group` as index.

Parameters

group	a GtkToolItemGroup
item	a GtkToolItem

Returns

the index of `item` in `group` or -1 if `item` is no child of `group`

Since: 2.20

---

gtk_tool_item_group_get_n_items ()

```
guint
gtk_tool_item_group_get_n_items (GtkToolItemGroup *group);
```

Gets the number of tool items in `group` .

Parameters

group	a GtkToolItemGroup

Returns

the number of tool items in `group`

Since: 2.20

---

gtk_tool_item_group_get_label ()

```
const gchar *
gtk_tool_item_group_get_label (GtkToolItemGroup *group);
```

Gets the label of `group` .

Parameters

group	a GtkToolItemGroup

Returns

the label of `group` . The label is an internal string of `group` and must not be modified. Note that `NULL` is returned if a custom label has been set with `gtk_tool_item_group_set_label_widget()`

Since: 2.20

---

gtk_tool_item_group_get_label_widget ()

```
GtkWidget *
gtk_tool_item_group_get_label_widget (GtkToolItemGroup *group);
```

Gets the label widget of `group` . See `gtk_tool_item_group_set_label_widget()`.

Parameters

group	a GtkToolItemGroup

Returns

the label widget of `group` .

[transfer none]

Since: 2.20

gtk_tool_item_group_get_nth_item ()

```
GtkToolItem *
gtk_tool_item_group_get_nth_item (GtkToolItemGroup *group,
                                  guint index);
```

Gets the tool item at `index` in group.

Parameters

group	a GtkToolItemGroup
index	the index

Returns

the GtkToolItem at index.

[transfer none]

Since: 2.20

---

gtk_tool_item_group_get_header_relief ()

```
GtkReliefStyle
gtk_tool_item_group_get_header_relief (GtkToolItemGroup *group);
```

Gets the relief mode of the header button of `group` .

Parameters

group	a GtkToolItemGroup

Returns

the GtkReliefStyle

Since: 2.20

---

gtk_tool_item_group_insert ()

```
void
gtk_tool_item_group_insert (GtkToolItemGroup *group,
                            GtkToolItem *item,
                            gint position);
```

Inserts `item` at `position` in the list of children of `group` .

Parameters

group	a GtkToolItemGroup
item	the GtkToolItem to insert into `group`
position	the position of `item` in `group` , starting with 0. The position -1 means end of list.

Since: 2.20

---

gtk_tool_item_group_new ()

```
GtkWidget *
gtk_tool_item_group_new (const gchar *label);
```

Creates a new tool item group with label `label` .

Parameters

label	the label of the new group

Returns

a new GtkToolItemGroup.

Since: 2.20

---

gtk_tool_item_group_set_collapsed ()

```
void
gtk_tool_item_group_set_collapsed (GtkToolItemGroup *group,
                                   gboolean collapsed);
```

Sets whether the `group` should be collapsed or expanded.

Parameters

group	a GtkToolItemGroup
collapsed	whether the `group` should be collapsed or expanded

Since: 2.20

---

gtk_tool_item_group_set_ellipsize ()

```
void
gtk_tool_item_group_set_ellipsize (GtkToolItemGroup *group,
                                   PangoEllipsizeMode ellipsize);
```

Sets the ellipsization mode which should be used by labels in `group` .

Parameters

group	a GtkToolItemGroup
ellipsize	the PangoEllipsizeMode labels in `group` should use

Since: 2.20

---

gtk_tool_item_group_set_item_position ()

```
void
gtk_tool_item_group_set_item_position (GtkToolItemGroup *group,
                                       GtkToolItem *item,
                                       gint position);
```

Sets the position of `item` in the list of children of `group` .

Parameters

group	a GtkToolItemGroup
item	the GtkToolItem to move to a new position, should be a child of `group` .
position	the new position of `item` in `group` , starting with 0. The position -1 means end of list.

Since: 2.20

---

gtk_tool_item_group_set_label ()

```
void
gtk_tool_item_group_set_label (GtkToolItemGroup *group,
                               const gchar *label);
```

Sets the label of the tool item group. The label is displayed in the header of the group.

Parameters

group	a GtkToolItemGroup
label	the new human-readable label of of the group

Since: 2.20

---

gtk_tool_item_group_set_label_widget ()

```
void
```

```
gtk_tool_item_group_set_label_widget (GtkToolItemGroup *group,
                                      GtkWidget *label_widget);
```

Sets the label of the tool item group. The label widget is displayed in the header of the group, in place of the usual label.

Parameters

group	a GtkToolItemGroup
label_widget	the widget to be displayed in place of the usual label

Since: 2.20

---

gtk_tool_item_group_set_header_relief ()

```
void
gtk_tool_item_group_set_header_relief (GtkToolItemGroup *group,
                                       GtkReliefStyle style);
```

Set the button relief of the group header. See `gtk_button_set_relief()` for details.

Parameters

group	a GtkToolItemGroup
style	the GtkReliefStyle

Since: 2.20

Types and Values

struct GtkToolItemGroup

```
struct GtkToolItemGroup;
```

This should not be accessed directly. Use the accessor functions below.

---

struct GtkToolItemGroupClass

```
struct GtkToolItemGroupClass {
  GtkContainerClass parent_class;
};
```

Members

Property Details

The “collapsed” property

```
“collapsed”                gboolean
```

Whether the group has been collapsed and items are hidden.

Flags: Read / Write

Default value: FALSE

The “ellipsize” property

```
“ellipsize”                PangoEllipsizeMode
```

Ellipsize for item group headers.

Flags: Read / Write

Default value: PANGO_ELLIPSIZE_NONE

The “header-relief” property

```
“header-relief”            GtkReliefStyle
```

Relief of the group header button.

Flags: Read / Write

Default value: GTK_RELIEF_NORMAL

The “label” property

```
“label”                    gchar *
```

The human-readable title of this item group.

Flags: Read / Write

Default value: ""

The “label-widget” property

```
“label-widget”             GtkWidget *
```

A widget to display in place of the usual label.

Flags: Read / Write

Child Property Details

The “expand” child property

“expand” gboolean

Whether the item should receive extra space when the group grows.

Flags: Read / Write

Default value: FALSE

The “fill” child property

“fill” gboolean

Whether the item should fill the available space.

Flags: Read / Write

Default value: TRUE

The “homogeneous” child property

“homogeneous” gboolean

Whether the item should be the same size as other homogeneous items.

Flags: Read / Write

Default value: TRUE

The “new-row” child property

“new-row” gboolean

Whether the item should start a new row.

Flags: Read / Write

Default value: FALSE

The “position” child property

“position” gint

Position of the item within this group.

Flags: Read / Write

Allowed values: >= 0

Default value: 0

Style Property Details

The "expander-size" style property

"expander-size" gint

Size of the expander arrow.

Flags: Read

Allowed values: >= 0

Default value: 16

The "header-spacing" style property

"header-spacing" gint

Spacing between expander arrow and caption.

Flags: Read

Allowed values: >= 0

Default value: 2

GtkSeparatorToolItem

GtkSeparatorToolItem — A toolbar item that separates groups of other toolbar items

Functions

GtkToolItem *	gtk_separator_tool_item_new ()
void	gtk_separator_tool_item_set_draw ()
gboolean	gtk_separator_tool_item_get_draw ()

Properties

gboolean	draw	Read / Write

Types and Values

struct	GtkSeparatorToolItem
struct	GtkSeparatorToolItemClass

Object Hierarchy

```
GObject
└── GInitiallyUnowned
```

```
        └── GtkWidget
            └── GtkContainer
                └── GtkBin
                    └── GtkToolItem
                        └── GtkSeparatorToolItem
```

Implemented Interfaces

GtkSeparatorToolItem implements AtkImplementorIface, GtkBuildable and GtkActivatable.

Includes

```
#include <gtk/gtk.h>
```

Description

A GtkSeparatorToolItem is a GtkToolItem that separates groups of other GtkToolItems. Depending on the theme, a GtkSeparatorToolItem will often look like a vertical line on horizontally docked toolbars.

If the GtkToolbar child property "expand" is `TRUE` and the property "draw" is `FALSE`, a GtkSeparatorToolItem will act as a "spring" that forces other items to the ends of the toolbar.

Use `gtk_separator_tool_item_new()` to create a new GtkSeparatorToolItem.

CSS nodes

GtkSeparatorToolItem has a single CSS node with name separator.

Functions

gtk_separator_tool_item_new ()

```
GtkToolItem *
gtk_separator_tool_item_new (void);
```

Create a new GtkSeparatorToolItem

Returns

the new GtkSeparatorToolItem

Since: 2.4

---

gtk_separator_tool_item_set_draw ()

```
void
gtk_separator_tool_item_set_draw (GtkSeparatorToolItem *item,
```

```
                                gboolean draw);
```

Whether `item` is drawn as a vertical line, or just blank. Setting this to `FALSE` along with `gtk_tool_item_set_expand()` is useful to create an item that forces following items to the end of the toolbar.

Parameters

item	a GtkSeparatorToolItem
draw	whether `item` is drawn as a vertical line

Since: 2.4

---

gtk_separator_tool_item_get_draw ()

```
gboolean
gtk_separator_tool_item_get_draw (GtkSeparatorToolItem *item);
```

Returns whether `item` is drawn as a line, or just blank. See `gtk_separator_tool_item_set_draw()`.

Parameters

item	a GtkSeparatorToolItem

Returns

`TRUE` if `item` is drawn as a line, or just blank.

Since: 2.4

Types and Values

struct GtkSeparatorToolItem

```
struct GtkSeparatorToolItem;
```

---

struct GtkSeparatorToolItemClass

```
struct GtkSeparatorToolItemClass {
  GtkToolItemClass parent_class;
};
```

Members

Property Details

The "draw" property

"draw" gboolean

Whether the separator is drawn, or just blank.

Flags: Read / Write

Default value: TRUE

See Also

GtkToolbar, GtkRadioToolButton

GtkToolButton

GtkToolButton — A GtkToolItem subclass that displays buttons

Functions

GtkToolItem *	gtk_tool_button_new ()
GtkToolItem *	gtk_tool_button_new_from_stock ()
void	gtk_tool_button_set_label ()
const gchar *	gtk_tool_button_get_label ()
void	gtk_tool_button_set_use_underline ()
gboolean	gtk_tool_button_get_use_underline ()
void	gtk_tool_button_set_stock_id ()
const gchar *	gtk_tool_button_get_stock_id ()
void	gtk_tool_button_set_icon_name ()
const gchar *	gtk_tool_button_get_icon_name ()
void	gtk_tool_button_set_icon_widget ()
GtkWidget *	gtk_tool_button_get_icon_widget ()
void	gtk_tool_button_set_label_widget ()
GtkWidget *	gtk_tool_button_get_label_widget ()

Properties

gchar *	icon-name	Read / Write
GtkWidget *	icon-widget	Read / Write
gchar *	label	Read / Write
GtkWidget *	label-widget	Read / Write
gchar *	stock-id	Read / Write
gboolean	use-underline	Read / Write

Style Properties

gint	icon-spacing	Read / Write

Signals

void	clicked	Action

Types and Values

struct	GtkToolButton
struct	GtkToolButtonClass

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkContainer
            └── GtkBin
                └── GtkToolItem
                    └── GtkToolButton
                        ├── GtkMenuToolButton
                        └── GtkToggleToolButton
```

Implemented Interfaces

GtkToolButton implements AtkImplementorIface, GtkBuildable, GtkActivatable and GtkActionable.

Includes

```
#include <gtk/gtk.h>
```

Description

GtkToolButtons are GtkToolItems containing buttons.

Use `gtk_tool_button_new()` to create a new GtkToolButton.

The label of a GtkToolButton is determined by the properties “label-widget”, “label”, and “stock-id”. If “label-widget” is non-`NULL`, then that widget is used as the label. Otherwise, if “label” is non-`NULL`, that string is used as the label. Otherwise, if “stock-id” is non-`NULL`, the label is determined by the stock item. Otherwise, the button does not have a label.

The icon of a GtkToolButton is determined by the properties “icon-widget” and “stock-id”. If “icon-widget” is non-`NULL`, then that widget is used as the icon. Otherwise, if “stock-id” is non-`NULL`, the icon is determined by the stock item. Otherwise, the button does not have a icon.

CSS nodes

GtkToolButton has a single CSS node with name toolbutton.

Functions

gtk_tool_button_new ()

```
GtkToolItem *
gtk_tool_button_new (GtkWidget *icon_widget,
                     const gchar *label);
```

Creates a new GtkToolButton using `icon_widget` as contents and `label` as label.

Parameters

label	a string that will be used as label, or `NULL`.	[allow-none]
icon_widget	a widget that will be used as the button contents, or `NULL`.	[allow-none]

Returns

A new GtkToolButton

Since: 2.4

---

gtk_tool_button_new_from_stock ()

```
GtkToolItem *
gtk_tool_button_new_from_stock (const gchar *stock_id);
```

`gtk_tool_button_new_from_stock` has been deprecated since version 3.10 and should not be used in newly-written code.

Use `gtk_tool_button_new()` together with `gtk_image_new_from_icon_name()` instead.

Creates a new GtkToolButton containing the image and text from a stock item. Some stock ids have preprocessor macros like GTK_STOCK_OK and GTK_STOCK_APPLY.

It is an error if `stock_id` is not a name of a stock item.

Parameters

stock_id	the name of the stock item

Returns

A new GtkToolButton

Since: 2.4

---

gtk_tool_button_set_label ()

```
void
gtk_tool_button_set_label (GtkToolButton *button,
                           const gchar *label);
```

Sets `label` as the label used for the tool button. The “label” property only has an effect if not overridden by a non-`NULL` “label-widget” property. If both the “label-widget” and “label” properties are `NULL`, the label is determined by the “stock-id” property. If the “stock-id” property is also `NULL`, `button` will not have a label.

Parameters

button	a GtkToolButton
label	a string that will be used as label, or [allow-none] `NULL`.

Since: 2.4

---

gtk_tool_button_get_label ()

```
const gchar *
gtk_tool_button_get_label (GtkToolButton *button);
```

Returns the label used by the tool button, or `NULL` if the tool button doesn’t have a label. or uses a the label from a stock item. The returned string is owned by GTK+, and must not be modified or freed.

Parameters

button	a GtkToolButton

Returns

The label, or `NULL`.

[nullable]

Since: 2.4

---

gtk_tool_button_set_use_underline ()

```
void
gtk_tool_button_set_use_underline (GtkToolButton *button,
                                   gboolean use_underline);
```

If set, an underline in the label property indicates that the next character should be used for the mnemonic accelerator key in the overflow menu. For example, if the label property is “_Open” and `use_underline` is `TRUE`, the label on the tool button will be “Open” and the item on the overflow menu will have an underlined

“O”.

Labels shown on tool buttons never have mnemonics on them; this property only affects the menu item on the overflow menu.

Parameters

button	a GtkToolButton
use_underline	whether the button label has the form “_Open”

Since: 2.4

---

gtk_tool_button_get_use_underline ()

```
gboolean
gtk_tool_button_get_use_underline (GtkToolButton *button);
```

Returns whether underscores in the label property are used as mnemonics on menu items on the overflow menu. See `gtk_tool_button_set_use_underline()`.

Parameters

button	a GtkToolButton

Returns

`TRUE` if underscores in the label property are used as mnemonics on menu items on the overflow menu.

Since: 2.4

---

gtk_tool_button_set_stock_id ()

```
void
gtk_tool_button_set_stock_id (GtkToolButton *button,
                              const gchar *stock_id);
```

`gtk_tool_button_set_stock_id` has been deprecated since version 3.10 and should not be used in newly-written code.

Use `gtk_tool_button_set_icon_name()` instead.

Sets the name of the stock item. See `gtk_tool_button_new_from_stock()`. The stock_id property only has an effect if not overridden by non-`NULL` “label-widget” and “icon-widget” properties.

Parameters

button	a GtkToolButton	
stock_id	a name of a stock item, or `NULL`.	[allow-none]

Since: 2.4

gtk_tool_button_get_stock_id ()

```
const gchar *
gtk_tool_button_get_stock_id (GtkToolButton *button);
```

`gtk_tool_button_get_stock_id` has been deprecated since version 3.10 and should not be used in newly-written code.

Use `gtk_tool_button_get_icon_name()` instead.

Returns the name of the stock item. See `gtk_tool_button_set_stock_id()`. The returned string is owned by GTK+ and must not be freed or modifed.

Parameters

button	a GtkToolButton	

Returns

the name of the stock item for `button` .

Since: 2.4

---

gtk_tool_button_set_icon_name ()

```
void
gtk_tool_button_set_icon_name (GtkToolButton *button,
                               const gchar *icon_name);
```

Sets the icon for the tool button from a named themed icon. See the docs for GtkIconTheme for more details. The “icon-name” property only has an effect if not overridden by non-`NULL` “label-widget”, “icon-widget” and “stock-id” properties.

Parameters

button	a GtkToolButton	
icon_name	the name of the themed icon.	[allow-none]

Since: 2.8

---

gtk_tool_button_get_icon_name ()

```
const gchar *
gtk_tool_button_get_icon_name (GtkToolButton *button);
```

Returns the name of the themed icon for the tool button, see `gtk_tool_button_set_icon_name()`.

Parameters

button	a GtkToolButton

Returns

the icon name or NULL if the tool button has no themed icon.

[nullable]

Since: 2.8

---

gtk_tool_button_set_icon_widget ()

```
void
gtk_tool_button_set_icon_widget (GtkToolButton *button,
                                 GtkWidget *icon_widget);
```

Sets icon as the widget used as icon on button . If icon_widget is NULL the icon is determined by the "stock-id" property. If the "stock-id" property is also NULL, button will not have an icon.

Parameters

button	a GtkToolButton	
icon_widget	the widget used as icon, or NULL.	[allow-none]

Since: 2.4

---

gtk_tool_button_get_icon_widget ()

```
GtkWidget *
gtk_tool_button_get_icon_widget (GtkToolButton *button);
```

Return the widget used as icon widget on button . See gtk_tool_button_set_icon_widget().

Parameters

button	a GtkToolButton

Returns

The widget used as icon on button , or NULL.

[nullable][transfer none]

Since: 2.4

---

gtk_tool_button_set_label_widget ()

```
void
gtk_tool_button_set_label_widget (GtkToolButton *button,
                                  GtkWidget *label_widget);
```

Sets `label_widget` as the widget that will be used as the label for `button` . If `label_widget` is `NULL` the "label" property is used as label. If "label" is also `NULL`, the label in the stock item determined by the "stock-id" property is used as label. If "stock-id" is also `NULL`, `button` does not have a label.

Parameters

button	a GtkToolButton	
label_widget	the widget used as label, or `NULL`.	[allow-none]

Since: 2.4

---

gtk_tool_button_get_label_widget ()

```
GtkWidget *
gtk_tool_button_get_label_widget (GtkToolButton *button);
```

Returns the widget used as label on `button` . See `gtk_tool_button_set_label_widget()`.

Parameters

button	a GtkToolButton

Returns

The widget used as label on `button` , or `NULL`.

[nullable][transfer none]

Since: 2.4

Types and Values

struct GtkToolButton

```
struct GtkToolButton;
```

---

struct GtkToolButtonClass

```
struct GtkToolButtonClass {
  GtkToolItemClass parent_class;

  GType button_type;
```

```
  /* signal */
  void       (* clicked)             (GtkToolButton    *tool_item);
};
```

Members

GType `button_type`;	
`clicked` ()	Signal emitted when the tool button is clicked with the mouse or activated with the keyboard.

Property Details

The `“icon-name”` property

```
  “icon-name”                gchar *
```

The name of the themed icon displayed on the item. This property only has an effect if not overridden by “label-widget”, “icon-widget” or “stock-id” properties.

Flags: Read / Write

Default value: NULL

Since: 2.8

The `“icon-widget”` property

```
  “icon-widget”              GtkWidget *
```

Icon widget to display in the item.

Flags: Read / Write

The `“label”` property

```
  “label”                    gchar *
```

Text to show in the item.

Flags: Read / Write

Default value: NULL

The `“label-widget”` property

```
  “label-widget”             GtkWidget *
```

Widget to use as the item label.

Flags: Read / Write

The “stock-id” property

“stock-id” gchar *

The stock icon displayed on the item.

GtkToolButton:stock-id has been deprecated since version 3.10 and should not be used in newly-written code.

Use “icon-name” instead.

Flags: Read / Write

Default value: NULL

The “use-underline” property

“use-underline” gboolean

If set, an underline in the label property indicates that the next character should be used for the mnemonic accelerator key in the overflow menu.

Flags: Read / Write

Default value: FALSE

Style Property Details

The “icon-spacing” style property

“icon-spacing” gint

Spacing in pixels between the icon and label.

Flags: Read / Write

Allowed values: >= 0

Default value: 3

Signal Details

The “clicked” signal

```
void
user_function (GtkToolButton *toolbutton,
               gpointer       user_data)
```

This signal is emitted when the tool button is clicked with the mouse or activated with the keyboard.

Parameters

toolbutton	the object that emitted the signal
user_data	user data set when the signal handler was connected.

Flags: Action

See Also

GtkToolbar, GtkMenuToolButton, GtkToggleToolButton, GtkRadioToolButton, GtkSeparatorToolItem

---

GtkMenuToolButton

GtkMenuToolButton — A GtkToolItem containing a button with an additional dropdown menu

Functions

GtkToolItem *	gtk_menu_tool_button_new ()
GtkToolItem *	gtk_menu_tool_button_new_from_stock ()
void	gtk_menu_tool_button_set_menu ()
GtkWidget *	gtk_menu_tool_button_get_menu ()
void	gtk_menu_tool_button_set_arrow_tooltip_text ()
void	gtk_menu_tool_button_set_arrow_tooltip_markup ()

Properties

GtkMenu *	menu	Read / Write

Signals

void	show-menu	Run First

Types and Values

struct	GtkMenuToolButton
struct	GtkMenuToolButtonClass

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkContainer
            └── GtkBin
                └── GtkToolItem
```

```
    └── GtkToolButton
        └── GtkMenuToolButton
```

Implemented Interfaces

GtkMenuToolButton implements AtkImplementorIface, GtkBuildable, GtkActivatable and GtkActionable.

Includes

```
#include <gtk/gtk.h>
```

Description

A GtkMenuToolButton is a GtkToolItem that contains a button and a small additional button with an arrow. When clicked, the arrow button pops up a dropdown menu.

Use `gtk_menu_tool_button_new()` to create a new GtkMenuToolButton.

GtkMenuToolButton as GtkBuildable

The GtkMenuToolButton implementation of the GtkBuildable interface supports adding a menu by specifying “menu” as the “type” attribute of a <child> element.

An example for a UI definition fragment with menus:

```
<object class="GtkMenuToolButton">
  <child type="menu">
    <object class="GtkMenu"/>
  </child>
</object>
```

Functions

gtk_menu_tool_button_new ()

```
GtkToolItem *
gtk_menu_tool_button_new (GtkWidget *icon_widget,
                          const gchar *label);
```

Creates a new GtkMenuToolButton using `icon_widget` as icon and `label` as label.

Parameters

icon_widget	a widget that will be used as icon widget, or `NULL`.	[allow-none]
label	a string that will be used as label, or `NULL`.	[allow-none]

Returns

the new GtkMenuToolButton

Since: 2.6

---

gtk_menu_tool_button_new_from_stock ()

```
GtkToolItem *
gtk_menu_tool_button_new_from_stock (const gchar *stock_id);
```

`gtk_menu_tool_button_new_from_stock` has been deprecated since version 3.10 and should not be used in newly-written code.

Use `gtk_menu_tool_button_new()` instead.

Creates a new GtkMenuToolButton. The new GtkMenuToolButton will contain an icon and label from the stock item indicated by `stock_id` .

Parameters

stock_id	the name of a stock item

Returns

the new GtkMenuToolButton

Since: 2.6

---

gtk_menu_tool_button_set_menu ()

```
void
gtk_menu_tool_button_set_menu (GtkMenuToolButton *button,
                               GtkWidget *menu);
```

Sets the GtkMenu that is popped up when the user clicks on the arrow. If `menu` is NULL, the arrow button becomes insensitive.

Parameters

button	a GtkMenuToolButton
menu	the GtkMenu associated with GtkMenuToolButton

Since: 2.6

---

gtk_menu_tool_button_get_menu ()

```
GtkWidget *
gtk_menu_tool_button_get_menu (GtkMenuToolButton *button);
```

Gets the GtkMenu associated with GtkMenuToolButton.

Parameters

button	a GtkMenuToolButton

Returns

the GtkMenu associated with GtkMenuToolButton.

[transfer none]

Since: 2.6

---

gtk_menu_tool_button_set_arrow_tooltip_text ()

```
void
gtk_menu_tool_button_set_arrow_tooltip_text
                                (GtkMenuToolButton *button,
                                 const gchar *text);
```

Sets the tooltip text to be used as tooltip for the arrow button which pops up the menu. See `gtk_tool_item_set_tooltip_text()` for setting a tooltip on the whole GtkMenuToolButton.

Parameters

button	a GtkMenuToolButton
text	text to be used as tooltip text for button's arrow button

Since: 2.12

---

gtk_menu_tool_button_set_arrow_tooltip_markup ()

```
void
gtk_menu_tool_button_set_arrow_tooltip_markup
                                (GtkMenuToolButton *button,
                                 const gchar *markup);
```

Sets the tooltip markup text to be used as tooltip for the arrow button which pops up the menu. See `gtk_tool_item_set_tooltip_text()` for setting a tooltip on the whole GtkMenuToolButton.

Parameters

button	a GtkMenuToolButton
markup	markup text to be used as tooltip text for button's arrow button

Since: 2.12

Types and Values

struct GtkMenuToolButton

```
struct GtkMenuToolButton;
```

---

struct GtkMenuToolButtonClass

```
struct GtkMenuToolButtonClass {
  GtkToolButtonClass parent_class;


  void (*show_menu) (GtkMenuToolButton *button);
};
```

Members

`show_menu` ()	Signal emitted before the menu is shown.

Property Details

The “menu” property

```
  “menu”                     GtkMenu *
```

The dropdown menu.

Flags: Read / Write

Signal Details

The “show-menu” signal

```
void
user_function (GtkMenuToolButton *button,
               gpointer           user_data)
```

The ::show-menu signal is emitted before the menu is shown.

It can be used to populate the menu on demand, using `gtk_menu_tool_button_set_menu()`.

Note that even if you populate the menu dynamically in this way, you must set an empty menu on the GtkMenuToolButton beforehand, since the arrow is made insensitive if the menu is not set.

Parameters

button	the object on which the signal is

	emitted
user_data	user data set when the signal handler was connected.

Flags: Run First

See Also

GtkToolbar, GtkToolButton

GtkToggleToolButton

GtkToggleToolButton — A GtkToolItem containing a toggle button

Functions

GtkToolItem *	gtk_toggle_tool_button_new ()
GtkToolItem *	gtk_toggle_tool_button_new_from_stock ()
void	gtk_toggle_tool_button_set_active ()
gboolean	gtk_toggle_tool_button_get_active ()

Properties

gboolean	active	Read / Write

Signals

void	toggled	Run First

Types and Values

struct	GtkToggleToolButton
struct	GtkToggleToolButtonClass

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkContainer
            └── GtkBin
                └── GtkToolItem
                    └── GtkToolButton
                        └── GtkToggleToolButton
                            └── GtkRadioToolButton
```

Implemented Interfaces

GtkToggleToolButton implements AtkImplementorIface, GtkBuildable, GtkActivatable and GtkActionable.

Includes

```
#include <gtk/gtk.h>
```

Description

A GtkToggleToolButton is a GtkToolItem that contains a toggle button.

Use `gtk_toggle_tool_button_new()` to create a new GtkToggleToolButton.

CSS nodes

GtkToggleToolButton has a single CSS node with name togglebutton.

Functions

gtk_toggle_tool_button_new ()

```
GtkToolItem *
gtk_toggle_tool_button_new (void);
```

Returns a new GtkToggleToolButton

Returns

a newly created GtkToggleToolButton

Since: 2.4

---

gtk_toggle_tool_button_new_from_stock ()

```
GtkToolItem *
gtk_toggle_tool_button_new_from_stock (const gchar *stock_id);
```

`gtk_toggle_tool_button_new_from_stock` has been deprecated since version 3.10 and should not be used in newly-written code.

Use `gtk_toggle_tool_button_new()` instead.

Creates a new GtkToggleToolButton containing the image and text from a stock item. Some stock ids have preprocessor macros like GTK_STOCK_OK and GTK_STOCK_APPLY.

It is an error if `stock_id` is not a name of a stock item.

Parameters

stock_id	the name of the stock item

Returns

A new GtkToggleToolButton

Since: 2.4

---

gtk_toggle_tool_button_set_active ()

```
void
gtk_toggle_tool_button_set_active (GtkToggleToolButton *button,
                                   gboolean is_active);
```

Sets the status of the toggle tool button. Set to TRUE if you want the GtkToggleButton to be “pressed in”, and FALSE to raise it. This action causes the toggled signal to be emitted.

Parameters

button	a GtkToggleToolButton
is_active	whether button should be active

Since: 2.4

---

gtk_toggle_tool_button_get_active ()

```
gboolean
gtk_toggle_tool_button_get_active (GtkToggleToolButton *button);
```

Queries a GtkToggleToolButton and returns its current state. Returns TRUE if the toggle button is pressed in and FALSE if it is raised.

Parameters

button	a GtkToggleToolButton

Returns

TRUE if the toggle tool button is pressed in, FALSE if not

Since: 2.4

Types and Values

struct GtkToggleToolButton

```
struct GtkToggleToolButton;
```

---

struct GtkToggleToolButtonClass

```
struct GtkToggleToolButtonClass {
  GtkToolButtonClass parent_class;


  /* signal */
  void (* toggled) (GtkToggleToolButton *button);
};
```

Members

`toggled` ()	Signal emitted whenever the toggle tool button changes state.

Property Details

The `"active"` property

```
  "active"                    gboolean
```

If the toggle tool button should be pressed in.

Flags: Read / Write

Default value: FALSE

Since: 2.8

Signal Details

The `"toggled"` signal

```
void
user_function (GtkToggleToolButton *toggle_tool_button,
               gpointer             user_data)
```

Emitted whenever the toggle tool button changes state.

Parameters

toggle_tool_button	the object that emitted the signal
user_data	user data set when the signal handler was connected.

Flags: Run First

See Also

GtkToolbar, GtkToolButton, GtkSeparatorToolItem

GtkRadioToolButton

GtkRadioToolButton — A toolbar item that contains a radio button

Functions

GtkToolItem *	gtk_radio_tool_button_new ()
GtkToolItem *	gtk_radio_tool_button_new_from_stock ()
GtkToolItem *	gtk_radio_tool_button_new_from_widget ()
GtkToolItem *	gtk_radio_tool_button_new_with_stock_from_widget ()
GSList *	gtk_radio_tool_button_get_group ()
void	gtk_radio_tool_button_set_group ()

Properties

GtkRadioToolButton *	group	Write

Types and Values

struct	GtkRadioToolButton

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkContainer
            └── GtkBin
                └── GtkToolItem
                    └── GtkToolButton
                        └── GtkToggleToolButton
                            └── GtkRadioToolButton
```

Implemented Interfaces

GtkRadioToolButton implements AtkImplementorIface, GtkBuildable, GtkActivatable and GtkActionable.

Includes

```
#include <gtk/gtk.h>
```

Description

A GtkRadioToolButton is a GtkToolItem that contains a radio button, that is, a button that is part of a group of toggle buttons where only one button can be active at a time.

Use `gtk_radio_tool_button_new()` to create a new GtkRadioToolButton. Use `gtk_radio_tool_button_new_from_widget()` to create a new GtkRadioToolButton that is part of the same group as an existing GtkRadioToolButton.

CSS nodes

GtkRadioToolButton has a single CSS node with name toolbutton.

Functions

gtk_radio_tool_button_new ()

```
GtkToolItem *
gtk_radio_tool_button_new (GSList *group);
```

Creates a new GtkRadioToolButton, adding it to `group` .

Parameters

group	An existing radio button group, or `NULL` if you are creating a new group.	[allow-none][element-type GtkRadioButton]

Returns

The new GtkRadioToolButton

Since: 2.4

---

gtk_radio_tool_button_new_from_stock ()

```
GtkToolItem *
gtk_radio_tool_button_new_from_stock (GSList *group,
                                      const gchar *stock_id);
```

`gtk_radio_tool_button_new_from_stock` has been deprecated since version 3.10 and should not be used in newly-written code.

Use `gtk_radio_tool_button_new()` instead.

Creates a new GtkRadioToolButton, adding it to `group` . The new GtkRadioToolButton will contain an icon and label from the stock item indicated by `stock_id` .

Parameters

group	an existing radio button group, or `NULL` if you are creating a new group.	[allow-none][element-type GtkRadioButton]
stock_id	the name of a stock item	

Returns

The new GtkRadioToolButton

Since: 2.4

---

gtk_radio_tool_button_new_from_widget ()

```
GtkToolItem *
gtk_radio_tool_button_new_from_widget (GtkRadioToolButton *group);
```

Creates a new GtkRadioToolButton adding it to the same group as `gruup`

[constructor]

Parameters

group	An existing GtkRadioToolButton, or `NULL`.	[allow-none]

Returns

The new GtkRadioToolButton.

[transfer none]

Since: 2.4

---

gtk_radio_tool_button_new_with_stock_from_widget ()

```
GtkToolItem *
gtk_radio_tool_button_new_with_stock_from_widget
                                (GtkRadioToolButton *group,
                                 const gchar *stock_id);
```

`gtk_radio_tool_button_new_with_stock_from_widget` has been deprecated since version 3.10 and should not be used in newly-written code.

gtk_radio_tool_button_new_from_widget

Creates a new GtkRadioToolButton adding it to the same group as `group` . The new GtkRadioToolButton will contain an icon and label from the stock item indicated by `stock_id` .

[constructor]

Parameters

group	An existing GtkRadioToolButton.	[allow-none]
stock_id	the name of a stock item	

Returns

A new GtkRadioToolButton.

[transfer none]

Since: 2.4

---

gtk_radio_tool_button_get_group ()

```
GSList *
gtk_radio_tool_button_get_group (GtkRadioToolButton *button);
```

Returns the radio button group `button` belongs to.

Parameters

button	a GtkRadioToolButton	

Returns

The group `button` belongs to.

[transfer none][element-type GtkRadioButton]

Since: 2.4

---

gtk_radio_tool_button_set_group ()

```
void
gtk_radio_tool_button_set_group (GtkRadioToolButton *button,
                                 GSList *group);
```

Adds `button` to `group` , removing it from the group it belonged to before.

Parameters

button	a GtkRadioToolButton	
group	an existing radio button group, or `NULL`.	[element-type GtkRadioButton] [allow-none]

Since: 2.4

Types and Values

struct GtkRadioToolButton

```
struct GtkRadioToolButton;
```

Property Details

The “group” property

```
  “group”                    GtkRadioToolButton *
```

Sets a new group for a radio tool button.

Flags: Write

Since: 2.4

See Also

GtkToolbar, GtkToolButton

GtkPopover

GtkPopover — Context dependent bubbles

Functions

GtkWidget *	gtk_popover_new ()
GtkWidget *	gtk_popover_new_from_model ()
void	gtk_popover_bind_model ()
void	gtk_popover_popup ()
void	gtk_popover_popdown ()
void	gtk_popover_set_relative_to ()
GtkWidget *	gtk_popover_get_relative_to ()
void	gtk_popover_set_pointing_to ()
gboolean	gtk_popover_get_pointing_to ()
void	gtk_popover_set_position ()
GtkPositionType	gtk_popover_get_position ()
void	gtk_popover_set_constrain_to ()
GtkPopoverConstraint	gtk_popover_get_constrain_to ()
void	gtk_popover_set_modal ()
gboolean	gtk_popover_get_modal ()
void	gtk_popover_set_transitions_enabled ()
gboolean	gtk_popover_get_transitions_enabled ()

void	gtk_popover_set_default_widget ()
GtkWidget *	gtk_popover_get_default_widget ()

Properties

GtkPopoverConstraint	constrain-to	Read / Write
gboolean	modal	Read / Write
GdkRectangle *	pointing-to	Read / Write
GtkPositionType	position	Read / Write
GtkWidget *	relative-to	Read / Write
gboolean	transitions-enabled	Read / Write

Signals

void	closed	Run Last

Types and Values

struct	GtkPopover
enum	GtkPopoverConstraint

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkContainer
            └── GtkBin
                └── GtkPopover
                    └── GtkPopoverMenu
```

Implemented Interfaces

GtkPopover implements AtkImplementorIface and GtkBuildable.

Includes

```
#include <gtk/gtk.h>
```

Description

GtkPopover is a bubble-like context window, primarily meant to provide context-dependent information or options. Popovers are attached to a widget, passed at construction time on `gtk_popover_new()`, or updated afterwards through `gtk_popover_set_relative_to()`, by default they will point to the whole widget area, although this behavior can be changed through `gtk_popover_set_pointing_to()`.

The position of a popover relative to the widget it is attached to can also be changed through `gtk_popover_set_position()`.

By default, GtkPopover performs a GTK+ grab, in order to ensure input events get redirected to it while it is

shown, and also so the popover is dismissed in the expected situations (clicks outside the popover, or the Esc key being pressed). If no such modal behavior is desired on a popover, `gtk_popover_set_modal()` may be called on it to tweak its behavior.

GtkPopover as menu replacement

GtkPopover is often used to replace menus. To facilitate this, it supports being populated from a GMenuModel, using `gtk_popover_new_from_model()`. In addition to all the regular menu model features, this function supports rendering sections in the model in a more compact form, as a row of icon buttons instead of menu items.

To use this rendering, set the ”display-hint” attribute of the section to ”horizontal-buttons” and set the icons of your items with the ”verb-icon” attribute.

```
<section>
  <attribute name="display-hint">horizontal-
buttons</attribute>
  <item>
    <attribute name="label">Cut</attribute>
    <attribute
name="action">app.cut</attribute>
    <attribute name="verb-icon">edit-cut-
symbolic</attribute>
  </item>
  <item>
    <attribute name="label">Copy</attribute>
    <attribute
name="action">app.copy</attribute>
    <attribute name="verb-icon">edit-copy-
symbolic</attribute>
  </item>
  <item>
    <attribute name="label">Paste</attribute>
    <attribute
name="action">app.paste</attribute>
    <attribute name="verb-icon">edit-paste-
symbolic</attribute>
  </item>
</section>
```

CSS nodes

GtkPopover has a single css node called popover. It always gets the .background style class and it gets the .menu style class if it is menu-like (e.g. GtkPopoverMenu or created using `gtk_popover_new_from_model()`.

Particular uses of GtkPopover, such as touch selection popups or magnifiers in GtkEntry or GtkTextView get style classes like .touch-selection or .magnifier to differentiate from plain popovers.

Functions

gtk_popover_new ()

```
GtkWidget *
```

```
gtk_popover_new (GtkWidget *relative_to);
```
Creates a new popover to point to `relative_to`

Parameters

relative_to	GtkWidget the popover is related to. [allow-none]

Returns

a new GtkPopover

Since: 3.12

---

gtk_popover_new_from_model ()

```
GtkWidget *
gtk_popover_new_from_model (GtkWidget *relative_to,
                            GMenuModel *model);
```
Creates a GtkPopover and populates it according to `model` . The popover is pointed to the `relative_to` widget.

The created buttons are connected to actions found in the GtkApplicationWindow to which the popover belongs - typically by means of being attached to a widget that is contained within the GtkApplicationWindows widget hierarchy.

Actions can also be added using `gtk_widget_insert_action_group()` on the menus attach widget or on any of its parent widgets.

Parameters

relative_to	GtkWidget the popover is related to. [allow-none]
model	a GMenuModel

Returns

the new GtkPopover

Since: 3.12

---

gtk_popover_bind_model ()

```
void
gtk_popover_bind_model (GtkPopover *popover,
                        GMenuModel *model,
                        const gchar *action_namespace);
```
Establishes a binding between a GtkPopover and a GMenuModel.

The contents of `popover` are removed and then refilled with menu items according to `model` . When `model` changes, `popover` is updated. Calling this function twice on `popover` with different `model` will cause the first binding to be replaced with a binding to the new model. If `model` is `NULL` then any previous binding is undone

and all children are removed.

If `action_namespace` is non-`NULL` then the effect is as if all actions mentioned in the `model` have their names prefixed with the namespace, plus a dot. For example, if the action “quit” is mentioned and `action_namespace` is “app” then the effective action name is “app.quit”.

This function uses GtkActionable to define the action name and target values on the created menu items. If you want to use an action group other than “app” and “win”, or if you want to use a GtkMenuShell outside of a GtkApplicationWindow, then you will need to attach your own action group to the widget hierarchy using `gtk_widget_insert_action_group()`. As an example, if you created a group with a “quit” action and inserted it with the name “mygroup” then you would use the action name “mygroup.quit” in your GMenuModel.

Parameters

popover	a GtkPopover
model	the GMenuModel to bind to or `NULL` to remove binding. [allow-none]
action_namespace	the namespace for actions in `model`. [allow-none]

Since: 3.12

---

gtk_popover_popup ()

```
void
gtk_popover_popup (GtkPopover *popover);
```

Pops `popover` up. This is different than a `gtk_widget_show()` call in that it shows the popover with a transition. If you want to show the popover without a transition, use `gtk_widget_show()`.

Parameters

popover	a GtkPopover

Since: 3.22

---

gtk_popover_popdown ()

```
void
gtk_popover_popdown (GtkPopover *popover);
```

Pops `popover` down.This is different than a `gtk_widget_hide()` call in that it shows the popover with a transition. If you want to hide the popover without a transition, use `gtk_widget_hide()`.

Parameters

popover	a GtkPopover

Since: 3.22

---

gtk_popover_set_relative_to ()

```
void
gtk_popover_set_relative_to (GtkPopover *popover,
                             GtkWidget *relative_to);
```

Sets a new widget to be attached to `popover` . If `popover` is visible, the position will be updated.

Note: the ownership of popovers is always given to their `relative_to` widget, so if `relative_to` is set to `NULL` on an attached `popover` , it will be detached from its previous widget, and consequently destroyed unless extra references are kept.

Parameters

popover	a GtkPopover	
relative_to	a GtkWidget.	[allow-none]

Since: 3.12

---

gtk_popover_get_relative_to ()

```
GtkWidget *
gtk_popover_get_relative_to (GtkPopover *popover);
```

Returns the widget `popover` is currently attached to

Parameters

popover	a GtkPopover

Returns

a GtkWidget.

[transfer none]

Since: 3.12

---

gtk_popover_set_pointing_to ()

```
void
gtk_popover_set_pointing_to (GtkPopover *popover,
                             const GdkRectangle *rect);
```

Sets the rectangle that `popover` will point to, in the coordinate space of the widget `popover` is attached to, see `gtk_popover_set_relative_to()`.

Parameters

popover	a GtkPopover
rect	rectangle to point to

Since: 3.12

---

gtk_popover_get_pointing_to ()

```
gboolean
gtk_popover_get_pointing_to (GtkPopover *popover,
                             GdkRectangle *rect);
```

If a rectangle to point to has been set, this function will return `TRUE` and fill in `rect` with such rectangle, otherwise it will return `FALSE` and fill in `rect` with the attached widget coordinates.

Parameters

popover	a GtkPopover	
rect	location to store the rectangle.	[out]

Returns

`TRUE` if a rectangle to point to was set.

---

gtk_popover_set_position ()

```
void
gtk_popover_set_position (GtkPopover *popover,
                          GtkPositionType position);
```

Sets the preferred position for `popover` to appear. If the `popover` is currently visible, it will be immediately updated.

This preference will be respected where possible, although on lack of space (eg. if close to the window edges), the GtkPopover may choose to appear on the opposite side

Parameters

popover	a GtkPopover
position	preferred popover position

Since: 3.12

---

gtk_popover_get_position ()

```
GtkPositionType
gtk_popover_get_position (GtkPopover *popover);
```

Returns the preferred position of `popover` .

Parameters

popover	a GtkPopover

Returns

The preferred position.

---

gtk_popover_set_constrain_to ()

```
void
gtk_popover_set_constrain_to (GtkPopover *popover,
                              GtkPopoverConstraint constraint);
```

Sets a constraint for positioning this popover.

Note that not all platforms support placing popovers freely, and may already impose constraints.

Parameters

popover	a GtkPopover
constraint	the new constraint

Since: 3.20

---

gtk_popover_get_constrain_to ()

```
GtkPopoverConstraint
gtk_popover_get_constrain_to (GtkPopover *popover);
```

Returns the constraint for placing this popover. See `gtk_popover_set_constrain_to()`.

Parameters

popover	a GtkPopover

Returns

the constraint for placing this popover.

Since: 3.20

---

gtk_popover_set_modal ()

```
void
gtk_popover_set_modal (GtkPopover *popover,
                       gboolean modal);
```

Sets whether `popover` is modal, a modal popover will grab all input within the toplevel and grab the keyboard focus on it when being displayed. Clicking outside the popover area or pressing Esc will dismiss the popover

and ungrab input.

Parameters

popover	a GtkPopover
modal	TRUE to make popover claim all input within the toplevel

Since: 3.12

---

gtk_popover_get_modal ()

```
gboolean
gtk_popover_get_modal (GtkPopover *popover);
```

Returns whether the popover is modal, see gtk_popover_set_modal to see the implications of this.

Parameters

popover	a GtkPopover

Returns

TRUE if `popover` is modal

Since: 3.12

---

gtk_popover_set_transitions_enabled ()

```
void
gtk_popover_set_transitions_enabled (GtkPopover *popover,
                                     gboolean transitions_enabled);
```

`gtk_popover_set_transitions_enabled` has been deprecated since version 3.22 and should not be used in newly-written code.

You can show or hide the popover without transitions using `gtk_widget_show()` and `gtk_widget_hide()` while `gtk_popover_popup()` and `gtk_popover_popdown()` will use transitions.

Sets whether show/hide transitions are enabled on this popover

Parameters

popover	a GtkPopover
transitions_enabled	Whether transitions are enabled

Since: 3.16

---

gtk_popover_get_transitions_enabled ()

```
gboolean
gtk_popover_get_transitions_enabled (GtkPopover *popover);
```

`gtk_popover_get_transitions_enabled` has been deprecated since version 3.22 and should not be used in newly-written code.

You can show or hide the popover without transitions using `gtk_widget_show()` and `gtk_widget_hide()` while `gtk_popover_popup()` and `gtk_popover_popdown()` will use transitions.

Returns whether show/hide transitions are enabled on this popover.

Parameters

popover	a GtkPopover	

Returns

TRUE if the show and hide transitions of the given popover are enabled, FALSE otherwise.

Since: 3.16

---

gtk_popover_set_default_widget ()

```
void
gtk_popover_set_default_widget (GtkPopover *popover,
                                GtkWidget *widget);
```

Sets the widget that should be set as default widget while the popover is shown (see `gtk_window_set_default()`). GtkPopover remembers the previous default widget and reestablishes it when the popover is dismissed.

Parameters

popover	a GtkPopover	
widget	the new default widget, or `NULL`.	[allow-none]

Since: 3.18

---

gtk_popover_get_default_widget ()

```
GtkWidget *
gtk_popover_get_default_widget (GtkPopover *popover);
```

Gets the widget that should be set as the default while the popover is shown.

Parameters

popover	a GtkPopover	

Returns

the default widget, or `NULL` if there is none.

[nullable][transfer none]

Since: 3.18

Types and Values

struct GtkPopover

```
struct GtkPopover;
```

---

enum GtkPopoverConstraint

Describes constraints to positioning of popovers. More values may be added to this enumeration in the future.

Members

GTK_POPOVER_CONSTRAINT_NONE	Don't constrain the popover position beyond what is imposed by the implementation
GTK_POPOVER_CONSTRAINT_WINDOW	Constrain the popover to the boundaries of the window that it is attached to

Since: 3.20

Property Details

The `“constrain-to”` property

```
  “constrain-to”             GtkPopoverConstraint
```

Sets a constraint for the popover position.

Flags: Read / Write

Default value: GTK_POPOVER_CONSTRAINT_WINDOW

Since: 3.20

The `“modal”` property

```
  “modal”                    gboolean
```

Sets whether the popover is modal (so other elements in the window do not receive input while the popover is visible).

Flags: Read / Write

Default value: TRUE

Since: 3.12

The “pointing-to” property

“pointing-to” GdkRectangle *

Marks a specific rectangle to be pointed.

Flags: Read / Write

Since: 3.12

The “position” property

“position” GtkPositionType

Sets the preferred position of the popover.

Flags: Read / Write

Default value: GTK_POS_TOP

Since: 3.12

The “relative-to” property

“relative-to” GtkWidget *

Sets the attached widget.

Flags: Read / Write

Since: 3.12

The “transitions-enabled” property

“transitions-enabled” gboolean

Whether show/hide transitions are enabled for this popover.

`GtkPopover:transitions-enabled` has been deprecated since version 3.22 and should not be used in newly-written code.

You can show or hide the popover without transitions using `gtk_widget_show()` and `gtk_widget_hide()` while `gtk_popover_popup()` and `gtk_popover_popdown()` will use transitions.

Flags: Read / Write

Default value: TRUE

Since: 3.16

Signal Details

The "closed" signal

```
void
user_function (GtkPopover *popover,
               gpointer    user_data)
```

This signal is emitted when the popover is dismissed either through API or user interaction.

Parameters

user_data	user data set when the signal handler was connected.

Flags: Run Last

Since: 3.12

GtkPopoverMenu

GtkPopoverMenu — Popovers to use as menus

Functions

GtkWidget *	gtk_popover_menu_new ()
void	gtk_popover_menu_open_submenu ()

Properties

gchar *	visible-submenu	Read / Write

Child Properties

gint	position	Read / Write
gchar *	submenu	Read / Write

Types and Values

GtkPopoverMenu

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkContainer
            └── GtkBin
                └── GtkPopover
                    └── GtkPopoverMenu
```

Implemented Interfaces

GtkPopoverMenu implements AtkImplementorIface and GtkBuildable.

Includes

```
#include <gtk/gtk.h>
```

Description

GtkPopoverMenu is a subclass of GtkPopover that treats its children like menus and allows switching between them. It is meant to be used primarily together with GtkModelButton, but any widget can be used, such as GtkSpinButton or GtkScale. In this respect, GtkPopoverMenu is more flexible than popovers that are created from a GMenuModel with `gtk_popover_new_from_model()`.

To add a child as a submenu, set the “submenu” child property to the name of the submenu. To let the user open this submenu, add a GtkModelButton whose “menu-name” property is set to the name you've given to the submenu.

By convention, the first child of a submenu should be a GtkModelButton to switch back to the parent menu. Such a button should use the “inverted” and “centered” properties to achieve a title-like appearance and place the submenu indicator at the opposite side. To switch back to the main menu, use "main" as the menu name.

Example

```
<object class="GtkPopoverMenu">
  <child>
    <object class="GtkBox">
      <property
name="visible">True</property>
      <property name="margin">10</property>
      <child>
        <object class="GtkModelButton">
          <property
name="visible">True</property>
          <property name="action-
name">win.frob</property>
          <property name="text"
translatable="yes">Frob</property>
        </object>
      </child>
      <child>
        <object class="GtkModelButton">
          <property
name="visible">True</property>
```

```
          <property
name="menu-name">more</property>
          <property name="text"
translatable="yes">More</property>
        </object>
      </child>
    </object>
  </child>
  <child>
    <object class="GtkBox">
      <property
name="visible">True</property>
      <property name="margin">10</property>
      <child>
        <object class="GtkModelButton">
          <property
name="visible">True</property>
          <property name="action-
name">win.foo</property>
          <property name="text"
translatable="yes">Foo</property>
        </object>
      </child>
      <child>
        <object class="GtkModelButton">
          <property
name="visible">True</property>
          <property name="action-
name">win.bar</property>
          <property name="text"
translatable="yes">Bar</property>
        </object>
      </child>
    </object>
    <packing>
      <property
name="submenu">more</property>
    </packing>
  </child>
</object>
```

Just like normal popovers created using gtk_popover_new_from_model, GtkPopoverMenu instances have a single css node called "popover" and get the .menu style class.

Functions

gtk_popover_menu_new ()

```
GtkWidget *
gtk_popover_menu_new (void);
```

Creates a new popover menu.

Returns

a new GtkPopoverMenu

Since: 3.16

gtk_popover_menu_open_submenu ()

```
void
gtk_popover_menu_open_submenu (GtkPopoverMenu *popover,
                               const gchar *name);
```

Opens a submenu of the `popover` . The `name` must be one of the names given to the submenus of `popover` with “submenu”, or "main" to switch back to the main menu.

GtkModelButton will open submenus automatically when the “menu-name” property is set, so this function is only needed when you are using other kinds of widgets to initiate menu changes.

Parameters

popover	a GtkPopoverMenu
name	the name of the menu to switch to

Since: 3.16

Types and Values

GtkPopoverMenu

```
typedef struct _GtkPopoverMenu GtkPopoverMenu;
```

Property Details

The “visible-submenu” property

```
  “visible-submenu”          gchar *
```

The name of the visible submenu.

Flags: Read / Write

Default value: NULL

Child Property Details

The “position” child property

```
  “position”                 gint
```

The index of the child in the parent.

Flags: Read / Write

Allowed values: >= -1

Default value: 0

The “submenu” child property

“submenu” gchar *

The submenu child property specifies the name of the submenu If it is NULL or "main", the child is used as the main menu, which is shown initially when the popover is mapped.

Flags: Read / Write

Default value: NULL

Since: 3.16

Selector Widgets and Dialogs

GtkColorChooser — Interface implemented by widgets for choosing colors

GtkColorButton — A button to launch a color selection dialog

GtkColorChooserWidget — A widget for choosing colors

GtkColorChooserDialog — A dialog for choosing colors

GtkFileChooser — File chooser interface used by GtkFileChooserWidget and GtkFileChooserDialog

GtkFileChooserButton — A button to launch a file selection dialog

GtkFileChooserNative — A native file chooser dialog, suitable for “File/Open” or “File/Save” commands

GtkFileChooserDialog — A file chooser dialog, suitable for “File/Open” or “File/Save” commands

GtkFileChooserWidget — A file chooser widget

GtkFileFilter — A filter for selecting a file subset

GtkFontChooser — Interface implemented by widgets displaying fonts

GtkFontButton — A button to launch a font chooser dialog

GtkFontChooserWidget — A widget for selecting fonts

GtkFontChooserDialog — A dialog for selecting fonts

GtkPlacesSidebar — Sidebar that displays frequently-used places in the file system

---

GtkColorChooser

GtkColorChooser — Interface implemented by widgets for choosing colors

Functions

void	gtk_color_chooser_get_rgba ()
void	gtk_color_chooser_set_rgba ()
gboolean	gtk_color_chooser_get_use_alpha ()
void	gtk_color_chooser_set_use_alpha ()
void	gtk_color_chooser_add_palette ()

Properties

GdkRGBA *	rgba	Read / Write
gboolean	use-alpha	Read / Write

Signals

void	color-activated	Run First

Types and Values

GtkColorChooser

Object Hierarchy

```
GInterface
└── GtkColorChooser
```

Prerequisites

GtkColorChooser requires GObject.

Known Implementations

GtkColorChooser is implemented by GtkColorButton, GtkColorChooserDialog and GtkColorChooserWidget.

Includes

```
#include <gtk/gtk.h>
```

Description

GtkColorChooser is an interface that is implemented by widgets for choosing colors. Depending on the situation, colors may be allowed to have alpha (translucency).

In GTK+, the main widgets that implement this interface are GtkColorChooserWidget, GtkColorChooserDialog and GtkColorButton.

Functions

gtk_color_chooser_get_rgba ()

```
void
gtk_color_chooser_get_rgba (GtkColorChooser *chooser,
                            GdkRGBA *color);
```

Gets the currently-selected color.

Parameters

chooser	a GtkColorChooser	
color	a GdkRGBA to fill in with the current color.	[out]

Since: 3.4

---

gtk_color_chooser_set_rgba ()

```
void
gtk_color_chooser_set_rgba (GtkColorChooser *chooser,
                            const GdkRGBA *color);
```

Sets the color.

Parameters

chooser	a GtkColorChooser
color	the new color

Since: 3.4

---

gtk_color_chooser_get_use_alpha ()

```
gboolean
gtk_color_chooser_get_use_alpha (GtkColorChooser *chooser);
```

Returns whether the color chooser shows the alpha channel.

Parameters

chooser	a GtkColorChooser

Returns

`TRUE` if the color chooser uses the alpha channel, `FALSE` if not

Since: 3.4

---

gtk_color_chooser_set_use_alpha ()

```
void
gtk_color_chooser_set_use_alpha (GtkColorChooser *chooser,
                                 gboolean use_alpha);
```

Sets whether or not the color chooser should use the alpha channel.

Parameters

chooser	a GtkColorChooser	
use_alpha	TRUE if color chooser should use alpha channel, FALSE if not	

Since: 3.4

---

gtk_color_chooser_add_palette ()

```
void
gtk_color_chooser_add_palette (GtkColorChooser *chooser,
                               GtkOrientation orientation,
                               gint colors_per_line,
                               gint n_colors,
                               GdkRGBA *colors);
```

Adds a palette to the color chooser. If `orientation` is horizontal, the colors are grouped in rows, with `colors_per_line` colors in each row. If `horizontal` is FALSE, the colors are grouped in columns instead.

The default color palette of GtkColorChooserWidget has 27 colors, organized in columns of 3 colors. The default gray palette has 9 grays in a single row.

The layout of the color chooser widget works best when the palettes have 9-10 columns.

Calling this function for the first time has the side effect of removing the default color and gray palettes from the color chooser.

If `colors` is NULL, removes all previously added palettes.

Parameters

chooser	a GtkColorChooser	
orientation	GTK_ORIENTATION_HORIZONTAL if the palette should be displayed in rows, GTK_ORIENTATION_VERTICAL for columns	
colors_per_line	the number of colors to show in each row/column	
n_colors	the total number of elements in `colors`	
colors	the colors of the palette, or NULL.	[allow-none][array length=n_colors]

Since: 3.4

Types and Values

GtkColorChooser

```
typedef struct _GtkColorChooser GtkColorChooser;
```

Property Details

The “rgba” property

```
  “rgba”                     GdkRGBA *
```

The ::rgba property contains the currently selected color, as a GdkRGBA struct. The property can be set to change the current selection programmatically.

Flags: Read / Write

Since: 3.4

The “use-alpha” property

```
  “use-alpha”                gboolean
```

When ::use-alpha is TRUE, colors may have alpha (translucency) information. When it is FALSE, the GdkRGBA struct obtained via the “rgba” property will be forced to have alpha == 1.

Implementations are expected to show alpha by rendering the color over a non-uniform background (like a checkerboard pattern).

Flags: Read / Write

Default value: TRUE

Since: 3.4

Signal Details

The “color-activated” signal

```
void
user_function (GtkColorChooser *chooser,
               GdkRGBA         *color,
               gpointer         user_data)
```

Emitted when a color is activated from the color chooser. This usually happens when the user clicks a color swatch, or a color is selected and the user presses one of the keys Space, Shift+Space, Return or Enter.

Parameters

chooser	the object which received the signal
color	the color
user_data	user data set when the signal handler was connected.

Flags: Run First

Since: 3.4

See Also

GtkColorChooserDialog, GtkColorChooserWidget, GtkColorButton

GtkColorButton

GtkColorButton — A button to launch a color selection dialog

Functions

GtkWidget *	gtk_color_button_new ()
GtkWidget *	gtk_color_button_new_with_color ()
GtkWidget *	gtk_color_button_new_with_rgba ()
void	gtk_color_button_set_color ()
void	gtk_color_button_get_color ()
void	gtk_color_button_set_alpha ()
guint16	gtk_color_button_get_alpha ()
void	gtk_color_button_set_rgba ()
void	gtk_color_button_get_rgba ()
void	gtk_color_button_set_use_alpha ()
gboolean	gtk_color_button_get_use_alpha ()
void	gtk_color_button_set_title ()
const gchar *	gtk_color_button_get_title ()

Properties

guint	alpha	Read / Write
GdkColor *	color	Read / Write
GdkRGBA *	rgba	Read / Write
gboolean	show-editor	Read / Write
gchar *	title	Read / Write
gboolean	use-alpha	Read / Write

Signals

void	color-set	Run First

Types and Values

struct	GtkColorButton

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkContainer
            └── GtkBin
                └── GtkButton
                    └── GtkColorButton
```

Implemented Interfaces

GtkColorButton implements AtkImplementorIface, GtkBuildable, GtkActionable, GtkActivatable and GtkColorChooser.

Includes

```
#include <gtk/gtk.h>
```

Description

The GtkColorButton is a button which displays the currently selected color and allows to open a color selection dialog to change the color. It is suitable widget for selecting a color in a preference dialog.

CSS nodes

GtkColorButton has a single CSS node with name button. To differentiate it from a plain GtkButton, it gets the .color style class.

Functions

gtk_color_button_new ()

```
GtkWidget *
gtk_color_button_new (void);
```

Creates a new color button.

This returns a widget in the form of a small button containing a swatch representing the current selected color. When the button is clicked, a color-selection dialog will open, allowing the user to select a color. The swatch will be updated to reflect the new color when the user finishes.

Returns

a new color button

Since: 2.4

---

gtk_color_button_new_with_color ()

```
GtkWidget *
gtk_color_button_new_with_color (const GdkColor *color);
```

`gtk_color_button_new_with_color` has been deprecated since version 3.4 and should not be used in newly-written code.

Use `gtk_color_button_new_with_rgba()` instead.

Creates a new color button.

Parameters

color	A GdkColor to set the current color with

Returns

a new color button

Since: 2.4

---

gtk_color_button_new_with_rgba ()

```
GtkWidget *
gtk_color_button_new_with_rgba (const GdkRGBA *rgba);
```

Creates a new color button.

Parameters

rgba	A GdkRGBA to set the current color with

Returns

a new color button

Since: 3.0

---

gtk_color_button_set_color ()

```
void
gtk_color_button_set_color (GtkColorButton *button,
                            const GdkColor *color);
```

`gtk_color_button_set_color` is deprecated and should not be used in newly-written code.

Use `gtk_color_chooser_set_rgba()` instead.

Sets the current color to be `color` .

Parameters

button	a GtkColorButton
color	A GdkColor to set the current color with

Since: 2.4

---

gtk_color_button_get_color ()

```
void
gtk_color_button_get_color (GtkColorButton *button,
                            GdkColor *color);
```

`gtk_color_button_get_color` has been deprecated since version 3.4 and should not be used in newly-written code.

Use `gtk_color_chooser_get_rgba()` instead.

Sets `color` to be the current color in the GtkColorButton widget.

Parameters

button	a GtkColorButton	
color	a GdkColor to fill in with the current color.	[out]

Since: 2.4

---

gtk_color_button_set_alpha ()

```
void
gtk_color_button_set_alpha (GtkColorButton *button,
                            guint16 alpha);
```

`gtk_color_button_set_alpha` has been deprecated since version 3.4 and should not be used in newly-written code.

Use `gtk_color_chooser_set_rgba()` instead.

Sets the current opacity to be `alpha` .

Parameters

button	a GtkColorButton
alpha	an integer between 0 and 65535

Since: 2.4

---

gtk_color_button_get_alpha ()

```
guint16
gtk_color_button_get_alpha (GtkColorButton *button);
```

`gtk_color_button_get_alpha` has been deprecated since version 3.4 and should not be used in newly-written code.

Use `gtk_color_chooser_get_rgba()` instead.

Returns the current alpha value.

Parameters

button	a GtkColorButton

Returns

an integer between 0 and 65535

Since: 2.4

---

gtk_color_button_set_rgba ()

```
void
gtk_color_button_set_rgba (GtkColorButton *button,
                           const GdkRGBA *rgba);
```

`gtk_color_button_set_rgba` has been deprecated since version 3.4 and should not be used in newly-written code.

Use `gtk_color_chooser_set_rgba()` instead.

Sets the current color to be `rgba` .

[skip]

Parameters

button	a GtkColorButton
rgba	a GdkRGBA to set the current color with

Since: 3.0

gtk_color_button_get_rgba ()

```
void
gtk_color_button_get_rgba (GtkColorButton *button,
                           GdkRGBA *rgba);
```

`gtk_color_button_get_rgba` has been deprecated since version 3.4 and should not be used in newly-written code.

Use `gtk_color_chooser_get_rgba()` instead.

Sets `rgba` to be the current color in the GtkColorButton widget.

[skip]

Parameters

button	a GtkColorButton	
rgba	a GdkRGBA to fill in with the current color.	[out]

Since: 3.0

---

gtk_color_button_set_use_alpha ()

```
void
gtk_color_button_set_use_alpha (GtkColorButton *button,
                                gboolean use_alpha);
```

`gtk_color_button_set_use_alpha` has been deprecated since version 3.4 and should not be used in newly-written code.

Use `gtk_color_chooser_set_use_alpha()` instead.

Sets whether or not the color button should use the alpha channel.

Parameters

button	a GtkColorButton
use_alpha	`TRUE` if color button should use alpha channel, `FALSE` if not

Since: 2.4

---

gtk_color_button_get_use_alpha ()

```
gboolean
gtk_color_button_get_use_alpha (GtkColorButton *button);
```

`gtk_color_button_get_use_alpha` has been deprecated since version 3.4 and should not be used in newly-written code.

Use `gtk_color_chooser_get_use_alpha()` instead.

Does the color selection dialog use the alpha channel ?

Parameters

button	a GtkColorButton

Returns

`TRUE` if the color sample uses alpha channel, `FALSE` if not

Since: 2.4

---

gtk_color_button_set_title ()

```
void
gtk_color_button_set_title (GtkColorButton *button,
                            const gchar *title);
```

Sets the title for the color selection dialog.

Parameters

button	a GtkColorButton
title	String containing new window title

Since: 2.4

---

gtk_color_button_get_title ()

```
const gchar *
gtk_color_button_get_title (GtkColorButton *button);
```

Gets the title of the color selection dialog.

Parameters

button	a GtkColorButton

Returns

An internal string, do not free the return value

Since: 2.4

Types and Values

struct GtkColorButton

```
struct GtkColorButton;
```

Property Details

The “alpha” property

```
  “alpha”                    guint
```

The selected opacity value (0 fully transparent, 65535 fully opaque).

Flags: Read / Write

Allowed values: <= 65535

Default value: 65535

Since: 2.4

The “color” property

```
  “color”                    GdkColor *
```

The selected color.

`GtkColorButton:color` has been deprecated since version 3.4 and should not be used in newly-written code.

Use “rgba” instead.

Flags: Read / Write

Since: 2.4

The “rgba” property

```
  “rgba”                     GdkRGBA *
```

The RGBA color.

Flags: Read / Write

Since: 3.0

The “show-editor” property

```
  “show-editor”              gboolean
```

Set this property to `TRUE` to skip the palette in the dialog and go directly to the color editor.

This property should be used in cases where the palette in the editor would be redundant, such as when the color button is already part of a palette.

Flags: Read / Write

Default value: FALSE

Since: 3.20

The “title” property

```
  “title”                    gchar *
```

The title of the color selection dialog

Flags: Read / Write

Default value: "Pick a Color"

Since: 2.4

The “use-alpha” property

```
  “use-alpha”                gboolean
```

If this property is set to TRUE, the color swatch on the button is rendered against a checkerboard background to show its opacity and the opacity slider is displayed in the color selection dialog.

Flags: Read / Write

Default value: FALSE

Since: 2.4

Signal Details

The “color-set” signal

```
void
user_function (GtkColorButton *widget,
               gpointer        user_data)
```

The ::color-set signal is emitted when the user selects a color. When handling this signal, use gtk_color_button_get_rgba() to find out which color was just selected.

Note that this signal is only emitted when the user changes the color. If you need to react to programmatic color changes as well, use the notify::color signal.

Parameters

widget	the object which received the signal.

user_data	user data set when the signal handler was connected.

Flags: Run First

Since: 2.4

See Also

GtkColorSelectionDialog, GtkFontButton

GtkColorChooserWidget

GtkColorChooserWidget — A widget for choosing colors

Functions

GtkWidget *	gtk_color_chooser_widget_new ()

Properties

gboolean	show-editor	Read / Write

Types and Values

struct	GtkColorChooserWidget
struct	GtkColorChooserWidgetClass

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkContainer
            └── GtkBox
                └── GtkColorChooserWidget
```

Implemented Interfaces

GtkColorChooserWidget implements AtkImplementorIface, GtkBuildable, GtkOrientable and GtkColorChooser.

Includes

```
#include <gtk/gtk.h>
```

Description

The GtkColorChooserWidget widget lets the user select a color. By default, the chooser presents a predefined palette of colors, plus a small number of settable custom colors. It is also possible to select a different color with the single-color editor. To enter the single-color editing mode, use the context menu of any color of the palette, or use the '+' button to add a new custom color.

The chooser automatically remembers the last selection, as well as custom colors.

To change the initially selected color, use `gtk_color_chooser_set_rgba()`. To get the selected color use `gtk_color_chooser_get_rgba()`.

The GtkColorChooserWidget is used in the GtkColorChooserDialog to provide a dialog for selecting colors.

CSS names

GtkColorChooserWidget has a single CSS node with name colorchooser.

Functions

gtk_color_chooser_widget_new ()

```
GtkWidget *
gtk_color_chooser_widget_new (void);
```

Creates a new GtkColorChooserWidget.

Returns

a new GtkColorChooserWidget

Since: 3.4

Types and Values

struct GtkColorChooserWidget

```
struct GtkColorChooserWidget;
```

---

struct GtkColorChooserWidgetClass

```
struct GtkColorChooserWidgetClass {
  GtkBoxClass parent_class;
};
```

Members

Property Details

The `“show-editor”` property

`“show-editor”` `gboolean`

The ::show-editor property is `TRUE` when the color chooser is showing the single-color editor. It can be set to switch the color chooser into single-color editing mode.

Flags: Read / Write

Default value: FALSE

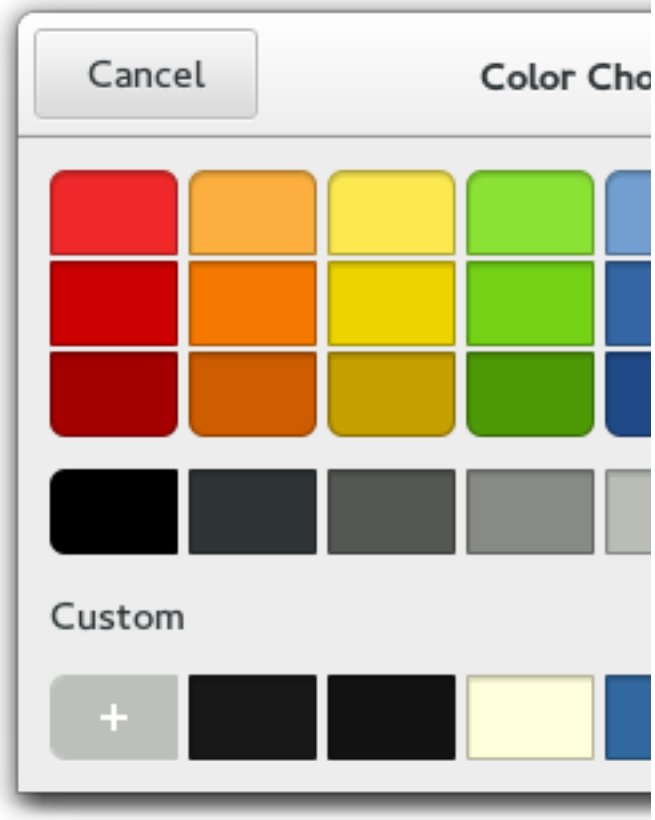
Since: 3.4

See Also

GtkColorChooserDialog

GtkColorChooserDialog

GtkColorChooserDialog — A dialog for choosing colors



Functions

GtkWidget *	gtk_color_chooser_dialog_new ()

Properties

gboolean	show-editor	Read / Write

Types and Values

struct	GtkColorChooserDialog

Object Hierarchy

```
    GObject
    └── GInitiallyUnowned
        └── GtkWidget
            └── GtkContainer
                └── GtkBin
                    └── GtkWindow
                        └── GtkDialog
                            └── GtkColorChooserDialog
```

Implemented Interfaces

GtkColorChooserDialog implements AtkImplementorIface, GtkBuildable and GtkColorChooser.

Includes

```
#include <gtk/gtk.h>
```

Description

The GtkColorChooserDialog widget is a dialog for choosing a color. It implements the GtkColorChooser interface.

Functions

gtk_color_chooser_dialog_new ()

```
GtkWidget *
gtk_color_chooser_dialog_new (const gchar *title,
                              GtkWindow *parent);
```

Creates a new GtkColorChooserDialog.

Parameters

title	Title of the dialog, or `NULL`.	[allow-none]
parent	Transient parent of the dialog, or `NULL`.	[allow-none]

Returns

a new GtkColorChooserDialog

Since: 3.4

Types and Values

struct GtkColorChooserDialog

```
struct GtkColorChooserDialog;
```

Property Details

The `"show-editor"` property

```
  "show-editor"              gboolean
```

Show editor.

Flags: Read / Write

Default value: FALSE

See Also

GtkColorChooser, GtkDialog

GtkFileChooser

GtkFileChooser — File chooser interface used by GtkFileChooserWidget and GtkFileChooserDialog

Functions

void	gtk_file_chooser_set_action ()
GtkFileChooserAction	gtk_file_chooser_get_action ()
void	gtk_file_chooser_set_local_only ()
gboolean	gtk_file_chooser_get_local_only ()
void	gtk_file_chooser_set_select_multiple ()

gboolean	gtk_file_chooser_get_select_multiple ()
void	gtk_file_chooser_set_show_hidden ()
gboolean	gtk_file_chooser_get_show_hidden ()
void	gtk_file_chooser_set_do_overwrite_confirmation ()
gboolean	gtk_file_chooser_get_do_overwrite_confirmation ()
void	gtk_file_chooser_set_create_folders ()
gboolean	gtk_file_chooser_get_create_folders ()
void	gtk_file_chooser_set_current_name ()
gchar *	gtk_file_chooser_get_current_name ()
gchar *	gtk_file_chooser_get_filename ()
gboolean	gtk_file_chooser_set_filename ()
gboolean	gtk_file_chooser_select_filename ()
void	gtk_file_chooser_unselect_filename ()
void	gtk_file_chooser_select_all ()
void	gtk_file_chooser_unselect_all ()
GSList *	gtk_file_chooser_get_filenames ()
gboolean	gtk_file_chooser_set_current_folder ()
gchar *	gtk_file_chooser_get_current_folder ()
gchar *	gtk_file_chooser_get_uri ()
gboolean	gtk_file_chooser_set_uri ()
gboolean	gtk_file_chooser_select_uri ()
void	gtk_file_chooser_unselect_uri ()
GSList *	gtk_file_chooser_get_uris ()
gboolean	gtk_file_chooser_set_current_folder_uri ()
gchar *	gtk_file_chooser_get_current_folder_uri ()
void	gtk_file_chooser_set_preview_widget ()
GtkWidget *	gtk_file_chooser_get_preview_widget ()
void	gtk_file_chooser_set_preview_widget_active ()
gboolean	gtk_file_chooser_get_preview_widget_active ()
void	gtk_file_chooser_set_use_preview_label ()
gboolean	gtk_file_chooser_get_use_preview_label ()
char *	gtk_file_chooser_get_preview_filename ()
char *	gtk_file_chooser_get_preview_uri ()
void	gtk_file_chooser_set_extra_widget ()
GtkWidget *	gtk_file_chooser_get_extra_widget ()
void	gtk_file_chooser_add_filter ()
void	gtk_file_chooser_remove_filter ()
GSList *	gtk_file_chooser_list_filters ()
void	gtk_file_chooser_set_filter ()
GtkFileFilter *	gtk_file_chooser_get_filter ()
gboolean	gtk_file_chooser_add_shortcut_folder ()
gboolean	gtk_file_chooser_remove_shortcut_folder ()
GSList *	gtk_file_chooser_list_shortcut_folders ()
gboolean	gtk_file_chooser_add_shortcut_folder_uri ()
gboolean	gtk_file_chooser_remove_shortcut_folder_uri ()
GSList *	gtk_file_chooser_list_shortcut_folder_uris ()
GFile *	gtk_file_chooser_get_current_folder_file ()
GFile *	gtk_file_chooser_get_file ()
GSList *	gtk_file_chooser_get_files ()
GFile *	gtk_file_chooser_get_preview_file ()
gboolean	gtk_file_chooser_select_file ()
gboolean	gtk_file_chooser_set_current_folder_file ()

gboolean	gtk_file_chooser_set_file ()
void	gtk_file_chooser_unselect_file ()

Properties

GtkFileChooserAction	action	Read / Write
gboolean	create-folders	Read / Write
gboolean	do-overwrite-confirmation	Read / Write
GtkWidget *	extra-widget	Read / Write
GtkFileFilter *	filter	Read / Write
gboolean	local-only	Read / Write
GtkWidget *	preview-widget	Read / Write
gboolean	preview-widget-active	Read / Write
gboolean	select-multiple	Read / Write
gboolean	show-hidden	Read / Write
gboolean	use-preview-label	Read / Write

Signals

GtkFileChooserConfirmation	confirm-overwrite	Run Last
void	current-folder-changed	Run Last
void	file-activated	Run Last
void	selection-changed	Run Last
void	update-preview	Run Last

Types and Values

	GtkFileChooser
enum	GtkFileChooserAction
enum	GtkFileChooserConfirmation
#define	GTK_FILE_CHOOSER_ERROR
enum	GtkFileChooserError

Object Hierarchy

```
GInterface
╰── GtkFileChooser
```

Prerequisites

GtkFileChooser requires GObject.

Known Implementations

GtkFileChooser is implemented by GtkFileChooserButton, GtkFileChooserDialog and GtkFileChooserWidget.

Includes

```
#include <gtk/gtk.h>
```

Description

GtkFileChooser is an interface that can be implemented by file selection widgets. In GTK+, the main objects that implement this interface are GtkFileChooserWidget, GtkFileChooserDialog, and GtkFileChooserButton. You do not need to write an object that implements the GtkFileChooser interface unless you are trying to adapt an existing file selector to expose a standard programming interface.

GtkFileChooser allows for shortcuts to various places in the filesystem. In the default implementation these are displayed in the left pane. It may be a bit confusing at first that these shortcuts come from various sources and in various flavours, so lets explain the terminology here:

- Bookmarks: are created by the user, by dragging folders from the right pane to the left pane, or by using the “Add”. Bookmarks can be renamed and deleted by the user.
- Shortcuts: can be provided by the application. For example, a Paint program may want to add a shortcut for a Clipart folder. Shortcuts cannot be modified by the user.
- Volumes: are provided by the underlying filesystem abstraction. They are the “roots” of the filesystem.

File Names and Encodings

When the user is finished selecting files in a GtkFileChooser, your program can get the selected names either as filenames or as URIs. For URIs, the normal escaping rules are applied if the URI contains non-ASCII characters. However, filenames are always returned in the character set specified by the `G_FILENAME_ENCODING` environment variable. Please see the GLib documentation for more details about this variable.

This means that while you can pass the result of `gtk_file_chooser_get_filename()` to `open()` or `fopen()`, you may not be able to directly set it as the text of a GtkLabel widget unless you convert it first to UTF-8, which all GTK+ widgets expect. You should use `g_filename_to_utf8()` to convert filenames into strings that can be passed to GTK+ widgets.

Adding a Preview Widget

You can add a custom preview widget to a file chooser and then get notification about when the preview needs to be updated. To install a preview widget, use `gtk_file_chooser_set_preview_widget()`. Then, connect to the “update-preview” signal to get notified when you need to update the contents of the preview.

Your callback should use `gtk_file_chooser_get_preview_filename()` to see what needs previewing. Once you have generated the preview for the corresponding file, you must call `gtk_file_chooser_set_preview_widget_active()` with a boolean flag that indicates whether your callback could successfully generate a preview.

Example: Using a Preview Widget

```
1                                                   {
2                                                     GtkImage *preview;
3
```

```
  ...

  preview = gtk_image_new ();

  gtk_file_chooser_set_preview_widget
(my_file_chooser, preview);
  g_signal_connect (my_file_chooser, "update-
preview",
                    G_CALLBACK
(update_preview_cb), preview);
}

static void
update_preview_cb (GtkFileChooser
*file_chooser, gpointer data)
{
  GtkWidget *preview;
  char *filename;
  GdkPixbuf *pixbuf;
  gboolean have_preview;

  preview = GTK_WIDGET (data);
  filename =
gtk_file_chooser_get_preview_filename
(file_chooser);

  pixbuf = gdk_pixbuf_new_from_file_at_size
(filename, 128, 128, NULL);
  have_preview = (pixbuf != NULL);
  g_free (filename);

  gtk_image_set_from_pixbuf (GTK_IMAGE
(preview), pixbuf);
  if (pixbuf)
    g_object_unref (pixbuf);

  gtk_file_chooser_set_preview_widget_active
(file_chooser, have_preview);
}
```

Adding Extra Widgets

You can add extra widgets to a file chooser to provide options that are not present in the default design. For example, you can add a toggle button to give the user the option to open a file in read-only mode. You can use `gtk_file_chooser_set_extra_widget()` to insert additional widgets in a file chooser.

An example for adding extra widgets:

```
GtkWidget *toggle;

...

toggle = gtk_check_button_new_with_label
("Open file read-only");
gtk_widget_show (toggle);
gtk_file_chooser_set_extra_widget
(my_file_chooser, toggle);
}
```

If you want to set more than one extra widget in the file chooser, you can a container such as a GtkBox or a GtkGrid and include your widgets in it. Then, set the container as the whole extra widget.

Functions

gtk_file_chooser_set_action ()

```
void
gtk_file_chooser_set_action (GtkFileChooser *chooser,
                             GtkFileChooserAction action);
```

Sets the type of operation that the chooser is performing; the user interface is adapted to suit the selected action. For example, an option to create a new folder might be shown if the action is `GTK_FILE_CHOOSER_ACTION_SAVE` but not if the action is `GTK_FILE_CHOOSER_ACTION_OPEN`.

Parameters

chooser	a GtkFileChooser
action	the action that the file selector is performing

Since: 2.4

---

gtk_file_chooser_get_action ()

```
GtkFileChooserAction
gtk_file_chooser_get_action (GtkFileChooser *chooser);
```

Gets the type of operation that the file chooser is performing; see `gtk_file_chooser_set_action()`.

Parameters

chooser	a GtkFileChooser

Returns

the action that the file selector is performing

Since: 2.4

---

gtk_file_chooser_set_local_only ()

```
void
gtk_file_chooser_set_local_only (GtkFileChooser *chooser,
                                 gboolean local_only);
```

Sets whether only local files can be selected in the file selector. If `local_only` is `TRUE` (the default), then the selected file or files are guaranteed to be accessible through the operating systems native file system and therefore the application only needs to worry about the filename functions in GtkFileChooser, like `gtk_file_chooser_get_filename()`, rather than the URI functions like `gtk_file_chooser_get_uri()`,

On some systems non-native files may still be available using the native filesystem via a userspace filesystem

(FUSE).

Parameters

chooser	a GtkFileChooser
local_only	TRUE if only local files can be selected

Since: 2.4

---

gtk_file_chooser_get_local_only ()

```
gboolean
gtk_file_chooser_get_local_only (GtkFileChooser *chooser);
```

Gets whether only local files can be selected in the file selector. See gtk_file_chooser_set_local_only()

Parameters

chooser	a GtkFileChooser

Returns

TRUE if only local files can be selected.

Since: 2.4

---

gtk_file_chooser_set_select_multiple ()

```
void
gtk_file_chooser_set_select_multiple (GtkFileChooser *chooser,
                                      gboolean select_multiple);
```

Sets whether multiple files can be selected in the file selector. This is only relevant if the action is set to be GTK_FILE_CHOOSER_ACTION_OPEN or GTK_FILE_CHOOSER_ACTION_SELECT_FOLDER.

Parameters

chooser	a GtkFileChooser
select_multiple	TRUE if multiple files can be selected.

Since: 2.4

---

gtk_file_chooser_get_select_multiple ()

```
gboolean
gtk_file_chooser_get_select_multiple (GtkFileChooser *chooser);
```

Gets whether multiple files can be selected in the file selector. See
gtk_file_chooser_set_select_multiple().

Parameters

chooser	a GtkFileChooser

Returns

TRUE if multiple files can be selected.

Since: 2.4

---

gtk_file_chooser_set_show_hidden ()

```
void
gtk_file_chooser_set_show_hidden (GtkFileChooser *chooser,
                                  gboolean show_hidden);
```

Sets whether hidden files and folders are displayed in the file selector.

Parameters

chooser	a GtkFileChooser
show_hidden	TRUE if hidden files and folders should be displayed.

Since: 2.6

---

gtk_file_chooser_get_show_hidden ()

```
gboolean
gtk_file_chooser_get_show_hidden (GtkFileChooser *chooser);
```

Gets whether hidden files and folders are displayed in the file selector. See
gtk_file_chooser_set_show_hidden().

Parameters

chooser	a GtkFileChooser

Returns

TRUE if hidden files and folders are displayed.

Since: 2.6

---

gtk_file_chooser_set_do_overwrite_confirmation ()

```
void
gtk_file_chooser_set_do_overwrite_confirmation
                                (GtkFileChooser *chooser,
                                 gboolean do_overwrite_confirmation);
```

Sets whether a file chooser in GTK_FILE_CHOOSER_ACTION_SAVE mode will present a confirmation dialog if the user types a file name that already exists. This is FALSE by default.

If set to TRUE, the chooser will emit the "confirm-overwrite" signal when appropriate.

If all you need is the stock confirmation dialog, set this property to TRUE. You can override the way confirmation is done by actually handling the "confirm-overwrite" signal; please refer to its documentation for the details.

Parameters

chooser	a GtkFileChooser
do_overwrite_confirmation	whether to confirm overwriting in save mode

Since: 2.8

---

gtk_file_chooser_get_do_overwrite_confirmation ()

```
gboolean
gtk_file_chooser_get_do_overwrite_confirmation
                                (GtkFileChooser *chooser);
```

Queries whether a file chooser is set to confirm for overwriting when the user types a file name that already exists.

Parameters

chooser	a GtkFileChooser

Returns

TRUE if the file chooser will present a confirmation dialog; FALSE otherwise.

Since: 2.8

---

gtk_file_chooser_set_create_folders ()

```
void
gtk_file_chooser_set_create_folders (GtkFileChooser *chooser,
                                     gboolean create_folders);
```

Sets whether file choser will offer to create new folders. This is only relevant if the action is not set to be GTK_FILE_CHOOSER_ACTION_OPEN.

Parameters

chooser	a GtkFileChooser	
create_folders	TRUE if the Create Folder button should be displayed	

Since: 2.18

---

gtk_file_chooser_get_create_folders ()

```
gboolean
gtk_file_chooser_get_create_folders (GtkFileChooser *chooser);
```

Gets whether file choser will offer to create new folders. See gtk_file_chooser_set_create_folders().

Parameters

chooser	a GtkFileChooser	

Returns

TRUE if the Create Folder button should be displayed.

Since: 2.18

---

gtk_file_chooser_set_current_name ()

```
void
gtk_file_chooser_set_current_name (GtkFileChooser *chooser,
                                   const gchar *name);
```

Sets the current name in the file selector, as if entered by the user. Note that the name passed in here is a UTF-8 string rather than a filename. This function is meant for such uses as a suggested name in a “Save As...” dialog. You can pass “Untitled.doc” or a similarly suitable suggestion for the name .

If you want to preselect a particular existing file, you should use gtk_file_chooser_set_filename() or gtk_file_chooser_set_uri() instead. Please see the documentation for those functions for an example of using gtk_file_chooser_set_current_name() as well.

Parameters

chooser	a GtkFileChooser	
name	the filename to use, as a UTF-8 string.	[type filename]

Since: 2.4

---

gtk_file_chooser_get_current_name ()

```
gchar *
gtk_file_chooser_get_current_name (GtkFileChooser *chooser);
```

Gets the current name in the file selector, as entered by the user in the text entry for “Name”.

This is meant to be used in save dialogs, to get the currently typed filename when the file itself does not exist yet. For example, an application that adds a custom extra widget to the file chooser for “file format” may want to change the extension of the typed filename based on the chosen format, say, from “.jpg” to “.png”.

Parameters

chooser	a GtkFileChooser

Returns

The raw text from the file chooser’s “Name” entry. Free this with `g_free()`. Note that this string is not a full pathname or URI; it is whatever the contents of the entry are. Note also that this string is in UTF-8 encoding, which is not necessarily the system’s encoding for filenames.

Since: 3.10

---

gtk_file_chooser_get_filename ()

```
gchar *
gtk_file_chooser_get_filename (GtkFileChooser *chooser);
```

Gets the filename for the currently selected file in the file selector. The filename is returned as an absolute path. If multiple files are selected, one of the filenames will be returned at random.

If the file chooser is in folder mode, this function returns the selected folder.

Parameters

chooser	a GtkFileChooser

Returns

The currently selected filename, or `NULL` if no file is selected, or the selected file can't be represented with a local filename. Free with `g_free()`.

[nullable][type filename]

Since: 2.4

---

gtk_file_chooser_set_filename ()

```
gboolean
gtk_file_chooser_set_filename (GtkFileChooser *chooser,
                               const char *filename);
```

Sets `filename` as the current filename for the file chooser, by changing to the file's parent folder and actually selecting the file in list; all other files will be unselected. If the `chooser` is in `GTK_FILE_CHOOSER_ACTION_SAVE` mode, the file's base name will also appear in the dialog's file name entry.

Note that the file must exist, or nothing will be done except for the directory change.

You should use this function only when implementing a save dialog for which you already have a file name to which the user may save. For example, when the user opens an existing file and then does Save As... to save a copy or a modified version. If you don't have a file name already — for example, if the user just created a new file and is saving it for the first time, do not call this function. Instead, use something similar to this:

```
1                                   if (document_is_new)
2                                     {
3                                       // the user just created a new document
4                                       gtk_file_chooser_set_current_name
5                                   (chooser, "Untitled document");
6                                     }
7                                   else
8                                     {
9                                       // the user edited an existing document
10                                      gtk_file_chooser_set_filename (chooser,
                                    existing_filename);
                                      }
```

In the first case, the file chooser will present the user with useful suggestions as to where to save his new file. In the second case, the file's existing location is already known, so the file chooser will use it.

Parameters

chooser	a GtkFileChooser	
filename	the filename to set as current.	[type filename]

Returns

Not useful.

Since: 2.4

---

gtk_file_chooser_select_filename ()

```
gboolean
gtk_file_chooser_select_filename (GtkFileChooser *chooser,
                                  const char *filename);
```

Selects a filename. If the file name isn't in the current folder of `chooser` , then the current folder of `chooser` will be changed to the folder containing `filename` .

Parameters

chooser	a GtkFileChooser	
filename	the filename to select.	[type filename]

Returns

Not useful.

See also: `gtk_file_chooser_set_filename()`

Since: 2.4

---

gtk_file_chooser_unselect_filename ()

```
void
gtk_file_chooser_unselect_filename (GtkFileChooser *chooser,
                                    const char *filename);
```

Unselects a currently selected filename. If the filename is not in the current directory, does not exist, or is otherwise not currently selected, does nothing.

Parameters

chooser	a GtkFileChooser	
filename	the filename to unselect.	[type filename]

Since: 2.4

---

gtk_file_chooser_select_all ()

```
void
gtk_file_chooser_select_all (GtkFileChooser *chooser);
```

Selects all the files in the current folder of a file chooser.

Parameters

chooser	a GtkFileChooser

Since: 2.4

---

gtk_file_chooser_unselect_all ()

```
void
gtk_file_chooser_unselect_all (GtkFileChooser *chooser);
```

Unselects all the files in the current folder of a file chooser.

Parameters

chooser	a GtkFileChooser

Since: 2.4

---

gtk_file_chooser_get_filenames ()

```
GSList *
gtk_file_chooser_get_filenames (GtkFileChooser *chooser);
```

Lists all the selected files and subfolders in the current folder of `chooser` . The returned names are full absolute paths. If files in the current folder cannot be represented as local filenames they will be ignored. (See `gtk_file_chooser_get_uris()`)

Parameters

chooser	a GtkFileChooser	

Returns

a GSList containing the filenames of all selected files and subfolders in the current folder. Free the returned list with `g_slist_free()`, and the filenames with `g_free()`.

[element-type filename][transfer full]

Since: 2.4

---

gtk_file_chooser_set_current_folder ()

```
gboolean
gtk_file_chooser_set_current_folder (GtkFileChooser *chooser,
                                     const gchar *filename);
```

Sets the current folder for `chooser` from a local filename. The user will be shown the full contents of the current folder, plus user interface elements for navigating to other folders.

In general, you should not use this function. See the section on setting up a file chooser dialog for the rationale behind this.

Parameters

chooser	a GtkFileChooser	
filename	the full path of the new current folder.	[type filename]

Returns

Not useful.

Since: 2.4

---

gtk_file_chooser_get_current_folder ()

```
gchar *
gtk_file_chooser_get_current_folder (GtkFileChooser *chooser);
```

Gets the current folder of `chooser` as a local filename. See `gtk_file_chooser_set_current_folder()`.

Note that this is the folder that the file chooser is currently displaying (e.g. "/home/username/Documents"), which is not the same as the currently-selected folder if the chooser is in `GTK_FILE_CHOOSER_ACTION_SELECT_FOLDER` mode (e.g. "/home/username/Documents/selected-folder/". To get the currently-selected folder in that mode, use `gtk_file_chooser_get_uri()` as the usual way to get the selection.

Parameters

chooser	a GtkFileChooser

Returns

the full path of the current folder, or `NULL` if the current path cannot be represented as a local filename. Free with `g_free()`. This function will also return `NULL` if the file chooser was unable to load the last folder that was requested from it; for example, as would be for calling `gtk_file_chooser_set_current_folder()` on a nonexistent folder.

[nullable][type filename]

Since: 2.4

---

gtk_file_chooser_get_uri ()

```
gchar *
gtk_file_chooser_get_uri (GtkFileChooser *chooser);
```

Gets the URI for the currently selected file in the file selector. If multiple files are selected, one of the filenames will be returned at random.

If the file chooser is in folder mode, this function returns the selected folder.

Parameters

chooser	a GtkFileChooser

Returns

The currently selected URI, or `NULL` if no file is selected. If `gtk_file_chooser_set_local_only()` is set to `TRUE` (the default) a local URI will be returned for any FUSE locations. Free with `g_free()`.

[nullable][transfer full]

Since: 2.4

---

gtk_file_chooser_set_uri ()

```
gboolean
```

```
gtk_file_chooser_set_uri (GtkFileChooser *chooser,
                          const char *uri);
```

Sets the file referred to by `uri` as the current file for the file chooser, by changing to the URI's parent folder and actually selecting the URI in the list. If the `chooser` is `GTK_FILE_CHOOSER_ACTION_SAVE` mode, the URI's base name will also appear in the dialog's file name entry.

Note that the URI must exist, or nothing will be done except for the directory change.

You should use this function only when implementing a save dialog for which you already have a file name to which the user may save. For example, when the user opens an existing file and then does Save As... to save a copy or a modified version. If you don't have a file name already — for example, if the user just created a new file and is saving it for the first time, do not call this function. Instead, use something similar to this:

```
1  if (document_is_new)
2    {
3      // the user just created a new document
4      gtk_file_chooser_set_current_name
5  (chooser, "Untitled document");
6    }
7  else
8    {
9      // the user edited an existing document
10     gtk_file_chooser_set_uri (chooser,
   existing_uri);
     }
```

In the first case, the file chooser will present the user with useful suggestions as to where to save his new file. In the second case, the file's existing location is already known, so the file chooser will use it.

Parameters

chooser	a GtkFileChooser
uri	the URI to set as current

Returns

Not useful.

Since: 2.4

---

gtk_file_chooser_select_uri ()

```
gboolean
gtk_file_chooser_select_uri (GtkFileChooser *chooser,
                             const char *uri);
```

Selects the file to by `uri` . If the URI doesn't refer to a file in the current folder of `chooser` , then the current folder of `chooser` will be changed to the folder containing `filename` .

Parameters

chooser	a GtkFileChooser
uri	the URI to select

Returns

Not useful.

Since: 2.4

---

gtk_file_chooser_unselect_uri ()

```
void
gtk_file_chooser_unselect_uri (GtkFileChooser *chooser,
                               const char *uri);
```

Unselects the file referred to by `uri` . If the file is not in the current directory, does not exist, or is otherwise not currently selected, does nothing.

Parameters

chooser	a GtkFileChooser
uri	the URI to unselect

Since: 2.4

---

gtk_file_chooser_get_uris ()

```
GSList *
gtk_file_chooser_get_uris (GtkFileChooser *chooser);
```

Lists all the selected files and subfolders in the current folder of `chooser` . The returned names are full absolute URIs.

Parameters

chooser	a GtkFileChooser

Returns

a GSList containing the URIs of all selected files and subfolders in the current folder. Free the returned list with `g_slist_free()`, and the filenames with `g_free()`.

[element-type utf8][transfer full]

Since: 2.4

---

gtk_file_chooser_set_current_folder_uri ()

```
gboolean
gtk_file_chooser_set_current_folder_uri
                               (GtkFileChooser *chooser,
                                const gchar *uri);
```

Sets the current folder for `chooser` from an URI. The user will be shown the full contents of the current folder, plus user interface elements for navigating to other folders.

In general, you should not use this function. See the section on setting up a file chooser dialog for the rationale behind this.

Parameters

chooser	a GtkFileChooser
uri	the URI for the new current folder

Returns

`TRUE` if the folder could be changed successfully, `FALSE` otherwise.

Since: 2.4

---

gtk_file_chooser_get_current_folder_uri ()

```
gchar *
gtk_file_chooser_get_current_folder_uri
                                (GtkFileChooser *chooser);
```

Gets the current folder of `chooser` as an URI. See `gtk_file_chooser_set_current_folder_uri()`.

Note that this is the folder that the file chooser is currently displaying (e.g. "file:///home/username/Documents"), which is not the same as the currently-selected folder if the chooser is in `GTK_FILE_CHOOSER_ACTION_SELECT_FOLDER` mode (e.g. "file:///home/username/Documents/selected-folder/". To get the currently-selected folder in that mode, use `gtk_file_chooser_get_uri()` as the usual way to get the selection.

Parameters

chooser	a GtkFileChooser

Returns

the URI for the current folder. Free with `g_free()`. This function will also return `NULL` if the file chooser was unable to load the last folder that was requested from it; for example, as would be for calling `gtk_file_chooser_set_current_folder_uri()` on a nonexistent folder.

[nullable][transfer full]

Since: 2.4

---

gtk_file_chooser_set_preview_widget ()

```
void
gtk_file_chooser_set_preview_widget (GtkFileChooser *chooser,
```

```
                                GtkWidget *preview_widget);
```

Sets an application-supplied widget to use to display a custom preview of the currently selected file. To implement a preview, after setting the preview widget, you connect to the "update-preview" signal, and call `gtk_file_chooser_get_preview_filename()` or `gtk_file_chooser_get_preview_uri()` on each change. If you can display a preview of the new file, update your widget and set the preview active using `gtk_file_chooser_set_preview_widget_active()`. Otherwise, set the preview inactive.

When there is no application-supplied preview widget, or the application-supplied preview widget is not active, the file chooser will display no preview at all.

Parameters

chooser	a GtkFileChooser
preview_widget	widget for displaying preview.

Since: 2.4

---

gtk_file_chooser_get_preview_widget ()

```
GtkWidget *
gtk_file_chooser_get_preview_widget (GtkFileChooser *chooser);
```

Gets the current preview widget; see `gtk_file_chooser_set_preview_widget()`.

Parameters

chooser	a GtkFileChooser

Returns

the current preview widget, or `NULL`.

[nullable][transfer none]

Since: 2.4

---

gtk_file_chooser_set_preview_widget_active ()

```
void
gtk_file_chooser_set_preview_widget_active
                                (GtkFileChooser *chooser,
                                 gboolean active);
```

Sets whether the preview widget set by `gtk_file_chooser_set_preview_widget()` should be shown for the current filename. When `active` is set to false, the file chooser may display an internally generated preview of the current file or it may display no preview at all. See `gtk_file_chooser_set_preview_widget()` for more details.

Parameters

chooser	a GtkFileChooser
active	whether to display the user-specified preview widget

Since: 2.4

---

gtk_file_chooser_get_preview_widget_active ()

```
gboolean
gtk_file_chooser_get_preview_widget_active
                                (GtkFileChooser *chooser);
```

Gets whether the preview widget set by `gtk_file_chooser_set_preview_widget()` should be shown for the current filename. See `gtk_file_chooser_set_preview_widget_active()`.

Parameters

chooser	a GtkFileChooser

Returns

`TRUE` if the preview widget is active for the current filename.

Since: 2.4

---

gtk_file_chooser_set_use_preview_label ()

```
void
gtk_file_chooser_set_use_preview_label
                                (GtkFileChooser *chooser,
                                 gboolean use_label);
```

Sets whether the file chooser should display a stock label with the name of the file that is being previewed; the default is `TRUE`. Applications that want to draw the whole preview area themselves should set this to `FALSE` and display the name themselves in their preview widget.

See also: `gtk_file_chooser_set_preview_widget()`

Parameters

chooser	a GtkFileChooser
use_label	whether to display a stock label with the name of the previewed file

Since: 2.4

---

gtk_file_chooser_get_use_preview_label ()

```
gboolean
gtk_file_chooser_get_use_preview_label
                                (GtkFileChooser *chooser);
```

Gets whether a stock label should be drawn with the name of the previewed file. See `gtk_file_chooser_set_use_preview_label()`.

Parameters

chooser	a GtkFileChooser

Returns

`TRUE` if the file chooser is set to display a label with the name of the previewed file, `FALSE` otherwise.

---

gtk_file_chooser_get_preview_filename ()

```
char *
gtk_file_chooser_get_preview_filename (GtkFileChooser *chooser);
```

Gets the filename that should be previewed in a custom preview widget. See `gtk_file_chooser_set_preview_widget()`.

Parameters

chooser	a GtkFileChooser

Returns

the filename to preview, or `NULL` if no file is selected, or if the selected file cannot be represented as a local filename. Free with `g_free()`.

[nullable][type filename]

Since: 2.4

---

gtk_file_chooser_get_preview_uri ()

```
char *
gtk_file_chooser_get_preview_uri (GtkFileChooser *chooser);
```

Gets the URI that should be previewed in a custom preview widget. See `gtk_file_chooser_set_preview_widget()`.

Parameters

chooser	a GtkFileChooser

Returns

the URI for the file to preview, or `NULL` if no file is selected. Free with `g_free()`.

[nullable][transfer full]

Since: 2.4

---

gtk_file_chooser_set_extra_widget ()

```
void
gtk_file_chooser_set_extra_widget (GtkFileChooser *chooser,
                                   GtkWidget *extra_widget);
```

Sets an application-supplied widget to provide extra options to the user.

Parameters

chooser	a GtkFileChooser
extra_widget	widget for extra options

Since: 2.4

---

gtk_file_chooser_get_extra_widget ()

```
GtkWidget *
gtk_file_chooser_get_extra_widget (GtkFileChooser *chooser);
```

Gets the current extra widget; see `gtk_file_chooser_set_extra_widget()`.

Parameters

chooser	a GtkFileChooser

Returns

the current extra widget, or `NULL`.

[nullable][transfer none]

Since: 2.4

---

gtk_file_chooser_add_filter ()

```
void
gtk_file_chooser_add_filter (GtkFileChooser *chooser,
                             GtkFileFilter *filter);
```

Adds `filter` to the list of filters that the user can select between. When a filter is selected, only files that are

passed by that filter are displayed.

Note that the `chooser` takes ownership of the filter, so you have to ref and sink it if you want to keep a reference.

Parameters

chooser	a GtkFileChooser	
filter	a GtkFileFilter.	[transfer full]

Since: 2.4

---

gtk_file_chooser_remove_filter ()

```
void
gtk_file_chooser_remove_filter (GtkFileChooser *chooser,
                                GtkFileFilter *filter);
```

Removes `filter` from the list of filters that the user can select between.

Parameters

chooser	a GtkFileChooser
filter	a GtkFileFilter

Since: 2.4

---

gtk_file_chooser_list_filters ()

```
GSList *
gtk_file_chooser_list_filters (GtkFileChooser *chooser);
```

Lists the current set of user-selectable filters; see `gtk_file_chooser_add_filter()`, `gtk_file_chooser_remove_filter()`.

Parameters

chooser	a GtkFileChooser

Returns

a GSList containing the current set of user selectable filters. The contents of the list are owned by GTK+, but you must free the list itself with `g_slist_free()` when you are done with it.

[element-type GtkFileFilter][transfer container]

Since: 2.4

---

gtk_file_chooser_set_filter ()

```
void
gtk_file_chooser_set_filter (GtkFileChooser *chooser,
                             GtkFileFilter *filter);
```

Sets the current filter; only the files that pass the filter will be displayed. If the user-selectable list of filters is non-empty, then the filter should be one of the filters in that list. Setting the current filter when the list of filters is empty is useful if you want to restrict the displayed set of files without letting the user change it.

Parameters

chooser	a GtkFileChooser
filter	a GtkFileFilter

Since: 2.4

---

gtk_file_chooser_get_filter ()

```
GtkFileFilter *
gtk_file_chooser_get_filter (GtkFileChooser *chooser);
```

Gets the current filter; see `gtk_file_chooser_set_filter()`.

Parameters

chooser	a GtkFileChooser

Returns

the current filter, or `NULL`.

[nullable][transfer none]

Since: 2.4

---

gtk_file_chooser_add_shortcut_folder ()

```
gboolean
gtk_file_chooser_add_shortcut_folder (GtkFileChooser *chooser,
                                      const char *folder,
                                      GError **error);
```

Adds a folder to be displayed with the shortcut folders in a file chooser. Note that shortcut folders do not get saved, as they are provided by the application. For example, you can use this to add a “/usr/share/mydrawprogram/Clipart” folder to the volume list.

Parameters

chooser	a GtkFileChooser	
folder	filename of the folder to add.	[type filename]

error	location to store error, or `NULL`.	[allow-none]

Returns

`TRUE` if the folder could be added successfully, `FALSE` otherwise. In the latter case, the `error` will be set as appropriate.

Since: 2.4

---

gtk_file_chooser_remove_shortcut_folder ()

```
gboolean
gtk_file_chooser_remove_shortcut_folder
                                (GtkFileChooser *chooser,
                                 const char *folder,
                                 GError **error);
```

Removes a folder from a file chooser's list of shortcut folders.

Parameters

chooser	a GtkFileChooser	
folder	filename of the folder to remove.	[type filename]
error	location to store error, or `NULL`.	[allow-none]

Returns

`TRUE` if the operation succeeds, `FALSE` otherwise. In the latter case, the `error` will be set as appropriate.

See also: `gtk_file_chooser_add_shortcut_folder()`

Since: 2.4

---

gtk_file_chooser_list_shortcut_folders ()

```
GSList *
gtk_file_chooser_list_shortcut_folders
                                (GtkFileChooser *chooser);
```

Queries the list of shortcut folders in the file chooser, as set by `gtk_file_chooser_add_shortcut_folder()`.

Parameters

chooser	a GtkFileChooser	

Returns

A list of folder filenames, or `NULL` if there are no shortcut folders. Free the returned list with `g_slist_free()`, and the filenames with `g_free()`.

[nullable][element-type filename][transfer full]

Since: 2.4

---

gtk_file_chooser_add_shortcut_folder_uri ()

```
gboolean
gtk_file_chooser_add_shortcut_folder_uri
                                (GtkFileChooser *chooser,
                                 const char *uri,
                                 GError **error);
```

Adds a folder URI to be displayed with the shortcut folders in a file chooser. Note that shortcut folders do not get saved, as they are provided by the application. For example, you can use this to add a “file:///usr/share/mydrawprogram/Clipart” folder to the volume list.

Parameters

chooser	a GtkFileChooser	
uri	URI of the folder to add	
error	location to store error, or `NULL`.	[allow-none]

Returns

`TRUE` if the folder could be added successfully, `FALSE` otherwise. In the latter case, the `error` will be set as appropriate.

Since: 2.4

---

gtk_file_chooser_remove_shortcut_folder_uri ()

```
gboolean
gtk_file_chooser_remove_shortcut_folder_uri
                                (GtkFileChooser *chooser,
                                 const char *uri,
                                 GError **error);
```

Removes a folder URI from a file chooser’s list of shortcut folders.

Parameters

chooser	a GtkFileChooser	
uri	URI of the folder to remove	
error	location to store error, or `NULL`.	[allow-none]

Returns

`TRUE` if the operation succeeds, `FALSE` otherwise. In the latter case, the `error` will be set as appropriate.

See also: `gtk_file_chooser_add_shortcut_folder_uri()`

Since: 2.4

---

gtk_file_chooser_list_shortcut_folder_uris ()

```
GSList *
gtk_file_chooser_list_shortcut_folder_uris
                                (GtkFileChooser *chooser);
```

Queries the list of shortcut folders in the file chooser, as set by `gtk_file_chooser_add_shortcut_folder_uri()`.

Parameters

chooser	a GtkFileChooser

Returns

A list of folder URIs, or `NULL` if there are no shortcut folders. Free the returned list with `g_slist_free()`, and the URIs with `g_free()`.

[nullable][element-type utf8][transfer full]

Since: 2.4

---

gtk_file_chooser_get_current_folder_file ()

```
GFile *
gtk_file_chooser_get_current_folder_file
                                (GtkFileChooser *chooser);
```

Gets the current folder of `chooser` as GFile. See `gtk_file_chooser_get_current_folder_uri()`.

Parameters

chooser	a GtkFileChooser

Returns

the GFile for the current folder.

[transfer full]

Since: 2.14

---

gtk_file_chooser_get_file ()

```
GFile *
gtk_file_chooser_get_file (GtkFileChooser *chooser);
```

Gets the GFile for the currently selected file in the file selector. If multiple files are selected, one of the files will be returned at random.

If the file chooser is in folder mode, this function returns the selected folder.

Parameters

chooser	a GtkFileChooser

Returns

a selected GFile. You own the returned file; use `g_object_unref()` to release it.

[transfer full]

Since: 2.14

---

gtk_file_chooser_get_files ()

```
GSList *
gtk_file_chooser_get_files (GtkFileChooser *chooser);
```

Lists all the selected files and subfolders in the current folder of `chooser` as GFile. An internal function, see `gtk_file_chooser_get_uris()`.

Parameters

chooser	a GtkFileChooser

Returns

a GSList containing a GFile for each selected file and subfolder in the current folder. Free the returned list with `g_slist_free()`, and the files with `g_object_unref()`.

[element-type GFile][transfer full]

Since: 2.14

---

gtk_file_chooser_get_preview_file ()

```
GFile *
gtk_file_chooser_get_preview_file (GtkFileChooser *chooser);
```

Gets the GFile that should be previewed in a custom preview Internal function, see `gtk_file_chooser_get_preview_uri()`.

Parameters

chooser	a GtkFileChooser

Returns

the GFile for the file to preview, or `NULL` if no file is selected. Free with `g_object_unref()`.

[nullable][transfer full]

Since: 2.14

---

gtk_file_chooser_select_file ()

```
gboolean
gtk_file_chooser_select_file (GtkFileChooser *chooser,
                              GFile *file,
                              GError **error);
```

Selects the file referred to by `file` . An internal function. See `_gtk_file_chooser_select_uri()`.

Parameters

chooser	a GtkFileChooser	
file	the file to select	
error	location to store error, or `NULL`.	[allow-none]

Returns

Not useful.

Since: 2.14

---

gtk_file_chooser_set_current_folder_file ()

```
gboolean
gtk_file_chooser_set_current_folder_file
                             (GtkFileChooser *chooser,
                              GFile *file,
                              GError **error);
```

Sets the current folder for `chooser` from a GFile. Internal function, see `gtk_file_chooser_set_current_folder_uri()`.

Parameters

chooser	a GtkFileChooser	
file	the GFile for the new folder	
error	location to store error, or `NULL`.	[allow-none]

Returns

`TRUE` if the folder could be changed successfully, `FALSE` otherwise.

Since: 2.14

---

gtk_file_chooser_set_file ()

```
gboolean
gtk_file_chooser_set_file (GtkFileChooser *chooser,
                           GFile *file,
                           GError **error);
```

Sets `file` as the current filename for the file chooser, by changing to the file's parent folder and actually selecting the file in list. If the `chooser` is in `GTK_FILE_CHOOSER_ACTION_SAVE` mode, the file's base name will also appear in the dialog's file name entry.

If the file name isn't in the current folder of `chooser` , then the current folder of `chooser` will be changed to the folder containing `filename` . This is equivalent to a sequence of `gtk_file_chooser_unselect_all()` followed by `gtk_file_chooser_select_filename()`.

Note that the file must exist, or nothing will be done except for the directory change.

If you are implementing a save dialog, you should use this function if you already have a file name to which the user may save; for example, when the user opens an existing file and then does Save As... If you don't have a file name already — for example, if the user just created a new file and is saving it for the first time, do not call this function. Instead, use something similar to this:

```
if (document_is_new)
  {
    // the user just created a new document
    gtk_file_chooser_set_current_folder_file
(chooser, default_file_for_saving);
    gtk_file_chooser_set_current_name
(chooser, "Untitled document");
  }
else
  {
    // the user edited an existing document
    gtk_file_chooser_set_file (chooser,
existing_file);
  }
```

Parameters

chooser	a GtkFileChooser	
file	the GFile to set as current	
error	location to store the error, or `NULL` to ignore errors.	[allow-none]

Returns

Not useful.

Since: 2.14

---

gtk_file_chooser_unselect_file ()

```
void
gtk_file_chooser_unselect_file (GtkFileChooser *chooser,
                                GFile *file);
```

Unselects the file referred to by `file` . If the file is not in the current directory, does not exist, or is otherwise not currently selected, does nothing.

Parameters

chooser	a GtkFileChooser
file	a GFile

Since: 2.14

Types and Values

GtkFileChooser

```
typedef struct _GtkFileChooser GtkFileChooser;
```

---

enum GtkFileChooserAction

Describes whether a GtkFileChooser is being used to open existing files or to save to a possibly new file.

Members

GTK_FILE_CHOOSER_ACTION_OPEN	Indicates open mode. The file chooser will only let the user pick an existing file.
GTK_FILE_CHOOSER_ACTION_SAVE	Indicates save mode. The file chooser will let the user pick an existing file, or type in a new filename.
GTK_FILE_CHOOSER_ACTION_SELECT_FOLDER	Indicates an Open mode for selecting folders. The file chooser will let the user pick an existing folder.
GTK_FILE_CHOOSER_ACTION_CREATE_FOLDER	Indicates a mode for creating a new folder. The file chooser will let the user name an existing or new folder.

---

enum GtkFileChooserConfirmation

Used as a return value of handlers for the "confirm-overwrite" signal of a GtkFileChooser. This value determines whether the file chooser will present the stock confirmation dialog, accept the user's choice of a filename, or let the user choose another filename.

Members

GTK_FILE_CHOOSER_CONFIRMATION_CONFIRM	The file chooser will present its stock dialog to confirm about overwriting an existing file.
GTK_FILE_CHOOSER_CONFIRMATION_ACCEPT_FILENAME	The file chooser will terminate and accept the user's choice of a file name.
GTK_FILE_CHOOSER_CONFIRMATION_SELECT_AGAIN	The file chooser will continue running, so as to let the user select another file name.

Since: 2.8

GTK_FILE_CHOOSER_ERROR

```
#define GTK_FILE_CHOOSER_ERROR (gtk_file_chooser_error_quark ())
```

Used to get the GError quark for GtkFileChooser errors.

---

enum GtkFileChooserError

These identify the various errors that can occur while calling GtkFileChooser functions.

Members

GTK_FILE_CHOOSER_ERROR_NONEXISTENT	Indicates that a file does not exist.
GTK_FILE_CHOOSER_ERROR_BAD_FILENAME	Indicates a malformed filename.
GTK_FILE_CHOOSER_ERROR_ALREADY_EXISTS	Indicates a duplicate path (e.g. when adding a bookmark).
GTK_FILE_CHOOSER_ERROR_INCOMPLETE_HOSTNAME	Indicates an incomplete hostname (e.g. "http://foo" without a slash after that).

Property Details

The “action” property

“action” GtkFileChooserAction

The type of operation that the file selector is performing.

Flags: Read / Write

Default value: GTK_FILE_CHOOSER_ACTION_OPEN

The “create-folders” property

“create-folders” gboolean

Whether a file chooser not in GTK_FILE_CHOOSER_ACTION_OPEN mode will offer the user to create new folders.

Flags: Read / Write

Default value: TRUE

Since: 2.18

The “do-overwrite-confirmation” property

“do-overwrite-confirmation” gboolean

Whether a file chooser in GTK_FILE_CHOOSER_ACTION_SAVE mode will present an overwrite confirmation dialog if the user selects a file name that already exists.

Flags: Read / Write

Default value: FALSE

Since: 2.8

The “extra-widget” property

“extra-widget” GtkWidget *

Application supplied widget for extra options.

Flags: Read / Write

The “filter” property

“filter” GtkFileFilter *

The current filter for selecting which files are displayed.

Flags: Read / Write

The “local-only” property

“local-only” gboolean

Whether the selected file(s) should be limited to local file: URLs.

Flags: Read / Write

Default value: TRUE

The “preview-widget” property

“preview-widget” GtkWidget *

Application supplied widget for custom previews.

Flags: Read / Write

The “preview-widget-active” property

“preview-widget-active” gboolean

Whether the application supplied widget for custom previews should be shown.

Flags: Read / Write

Default value: TRUE

The “select-multiple” property

“select-multiple” gboolean

Whether to allow multiple files to be selected.

Flags: Read / Write

Default value: FALSE

The “show-hidden” property

“show-hidden” gboolean

Whether the hidden files and folders should be displayed.

Flags: Read / Write

Default value: FALSE

The “use-preview-label” property

“use-preview-label” gboolean

Whether to display a stock label with the name of the previewed file.

Flags: Read / Write

Default value: TRUE

Signal Details

The `"confirm-overwrite"` signal

```
GtkFileChooserConfirmation
user_function (GtkFileChooser *chooser,
               gpointer        user_data)
```

This signal gets emitted whenever it is appropriate to present a confirmation dialog when the user has selected a file name that already exists. The signal only gets emitted when the file chooser is in `GTK_FILE_CHOOSER_ACTION_SAVE` mode.

Most applications just need to turn on the "do-overwrite-confirmation" property (or call the `gtk_file_chooser_set_do_overwrite_confirmation()` function), and they will automatically get a stock confirmation dialog. Applications which need to customize this behavior should do that, and also connect to the "confirm-overwrite" signal.

A signal handler for this signal must return a GtkFileChooserConfirmation value, which indicates the action to take. If the handler determines that the user wants to select a different filename, it should return `GTK_FILE_CHOOSER_CONFIRMATION_SELECT_AGAIN`. If it determines that the user is satisfied with his choice of file name, it should return `GTK_FILE_CHOOSER_CONFIRMATION_ACCEPT_FILENAME`. On the other hand, if it determines that the stock confirmation dialog should be used, it should return `GTK_FILE_CHOOSER_CONFIRMATION_CONFIRM`. The following example illustrates this.

Custom confirmation

```
static GtkFileChooserConfirmation
confirm_overwrite_callback (GtkFileChooser
*chooser, gpointer data)
{
  char *uri;

  uri = gtk_file_chooser_get_uri (chooser);

  if (is_uri_read_only (uri))
    {
      if
(user_wants_to_replace_read_only_file (uri))
        return
GTK_FILE_CHOOSER_CONFIRMATION_ACCEPT_FILENAME
;
      else
        return
GTK_FILE_CHOOSER_CONFIRMATION_SELECT_AGAIN;
    } else
      return
GTK_FILE_CHOOSER_CONFIRMATION_CONFIRM; //
fall back to the default dialog
}
```

```
...

chooser = gtk_file_chooser_dialog_new (...);

gtk_file_chooser_set_do_overwrite_confirmation (GTK_FILE_CHOOSER (dialog), TRUE);
g_signal_connect (chooser, "confirm-overwrite",
                  G_CALLBACK (confirm_overwrite_callback), NULL);

if (gtk_dialog_run (chooser) == GTK_RESPONSE_ACCEPT)
        save_to_file (gtk_file_chooser_get_filename (GTK_FILE_CHOOSER (chooser));

gtk_widget_destroy (chooser);
```

Parameters

chooser	the object which received the signal.
user_data	user data set when the signal handler was connected.

Returns

a GtkFileChooserConfirmation value that indicates which action to take after emitting the signal.

Flags: Run Last

Since: 2.8

The `"current-folder-changed"` signal

```
void
user_function (GtkFileChooser *chooser,
               gpointer        user_data)
```

This signal is emitted when the current folder in a GtkFileChooser changes. This can happen due to the user performing some action that changes folders, such as selecting a bookmark or visiting a folder on the file list. It can also happen as a result of calling a function to explicitly change the current folder in a file chooser.

Normally you do not need to connect to this signal, unless you need to keep track of which folder a file chooser is showing.

See also: `gtk_file_chooser_set_current_folder()`, `gtk_file_chooser_get_current_folder()`, `gtk_file_chooser_set_current_folder_uri()`, `gtk_file_chooser_get_current_folder_uri()`.

Parameters

chooser	the object which received the signal.
user_data	user data set when the signal

handler was connected.

Flags: Run Last

The "file-activated" signal

```
void
user_function (GtkFileChooser *chooser,
               gpointer        user_data)
```

This signal is emitted when the user "activates" a file in the file chooser. This can happen by double-clicking on a file in the file list, or by pressing `Enter`.

Normally you do not need to connect to this signal. It is used internally by GtkFileChooserDialog to know when to activate the default button in the dialog.

See also: `gtk_file_chooser_get_filename()`, `gtk_file_chooser_get_filenames()`, `gtk_file_chooser_get_uri()`, `gtk_file_chooser_get_uris()`.

Parameters

chooser	the object which received the signal.
user_data	user data set when the signal handler was connected.

Flags: Run Last

The "selection-changed" signal

```
void
user_function (GtkFileChooser *chooser,
               gpointer        user_data)
```

This signal is emitted when there is a change in the set of selected files in a GtkFileChooser. This can happen when the user modifies the selection with the mouse or the keyboard, or when explicitly calling functions to change the selection.

Normally you do not need to connect to this signal, as it is easier to wait for the file chooser to finish running, and then to get the list of selected files using the functions mentioned below.

See also: `gtk_file_chooser_select_filename()`, `gtk_file_chooser_unselect_filename()`, `gtk_file_chooser_get_filename()`, `gtk_file_chooser_get_filenames()`, `gtk_file_chooser_select_uri()`, `gtk_file_chooser_unselect_uri()`, `gtk_file_chooser_get_uri()`, `gtk_file_chooser_get_uris()`.

Parameters

chooser	the object which received the signal.
user_data	user data set when the signal handler was connected.

Flags: Run Last

The "update-preview" signal

```
void
user_function (GtkFileChooser *chooser,
               gpointer        user_data)
```

This signal is emitted when the preview in a file chooser should be regenerated. For example, this can happen when the currently selected file changes. You should use this signal if you want your file chooser to have a preview widget.

Once you have installed a preview widget with `gtk_file_chooser_set_preview_widget()`, you should update it when this signal is emitted. You can use the functions `gtk_file_chooser_get_preview_filename()` or `gtk_file_chooser_get_preview_uri()` to get the name of the file to preview. Your widget may not be able to preview all kinds of files; your callback must call `gtk_file_chooser_set_preview_widget_active()` to inform the file chooser about whether the preview was generated successfully or not.

Please see the example code in Using a Preview Widget.

See also: `gtk_file_chooser_set_preview_widget()`, `gtk_file_chooser_set_preview_widget_active()`, `gtk_file_chooser_set_use_preview_label()`, `gtk_file_chooser_get_preview_filename()`, `gtk_file_chooser_get_preview_uri()`.

Parameters

chooser	the object which received the signal.
user_data	user data set when the signal handler was connected.

Flags: Run Last

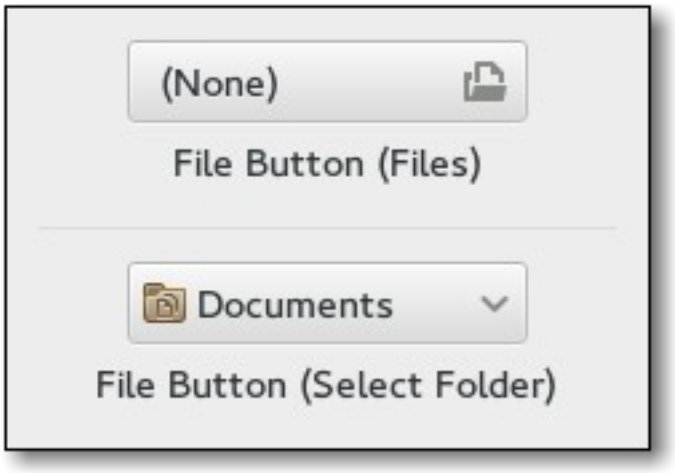
See Also

GtkFileChooserDialog, GtkFileChooserWidget, GtkFileChooserButton

---

GtkFileChooserButton

GtkFileChooserButton — A button to launch a file selection dialog



Functions

GtkWidget *	gtk_file_chooser_button_new ()
GtkWidget *	gtk_file_chooser_button_new_with_dialog ()
const gchar *	gtk_file_chooser_button_get_title ()
void	gtk_file_chooser_button_set_title ()
gint	gtk_file_chooser_button_get_width_chars ()
void	gtk_file_chooser_button_set_width_chars ()
gboolean	gtk_file_chooser_button_get_focus_on_click ()
void	gtk_file_chooser_button_set_focus_on_click ()

Properties

GtkFileChooser *	dialog	Write / Construct Only
gchar *	title	Read / Write
gint	width-chars	Read / Write

Signals

void	file-set	Run First

Types and Values

struct	GtkFileChooserButton
struct	GtkFileChooserButtonClass

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkContainer
            └── GtkBox
                └── GtkFileChooserButton
```

Implemented Interfaces

GtkFileChooserButton implements AtkImplementorIface, GtkBuildable, GtkOrientable and GtkFileChooser.

Includes

```
#include <gtk/gtk.h>
```

Description

The GtkFileChooserButton is a widget that lets the user select a file. It implements the GtkFileChooser interface. Visually, it is a file name with a button to bring up a GtkFileChooserDialog. The user can then use that dialog to change the file associated with that button. This widget does not support setting the “select-multiple” property to `TRUE`.

Create a button to let the user select a file in /etc

```
1                                                   {
2                                                     GtkWidget *button;
3
4                                                     button = gtk_file_chooser_button_new
5                                                   (_("Select a file"),
6
7                                                   GTK_FILE_CHOOSER_ACTION_OPEN);
8                                                     gtk_file_chooser_set_current_folder
                                                    (GTK_FILE_CHOOSER (button),

                                                    "/etc");
                                                    }
```

The GtkFileChooserButton supports the GtkFileChooserActions `GTK_FILE_CHOOSER_ACTION_OPEN` and `GTK_FILE_CHOOSER_ACTION_SELECT_FOLDER`.

> The GtkFileChooserButton will ellipsize the label, and will thus request little horizontal space. To give the button more space, you should call `gtk_widget_get_preferred_size()`, `gtk_file_chooser_button_set_width_chars()`, or pack the button in such a way that other interface elements give space to the widget.

CSS nodes

GtkFileChooserButton has a CSS node with name “filechooserbutton”, containing a subnode for the internal button with name “button” and style class “.file”.

Functions

gtk_file_chooser_button_new ()

```
GtkWidget *
gtk_file_chooser_button_new (const gchar *title,
                             GtkFileChooserAction action);
```

Creates a new file-selecting button widget.

Parameters

title	the title of the browse dialog.
action	the open mode for the widget.

Returns

a new button widget.

Since: 2.6

gtk_file_chooser_button_new_with_dialog ()

```
GtkWidget *
gtk_file_chooser_button_new_with_dialog
                                (GtkWidget *dialog);
```

Creates a GtkFileChooserButton widget which uses `dialog` as its file-picking window.

Note that `dialog` must be a GtkDialog (or subclass) which implements the GtkFileChooser interface and must not have `GTK_DIALOG_DESTROY_WITH_PARENT` set.

Also note that the dialog needs to have its confirmative button added with response `GTK_RESPONSE_ACCEPT` or `GTK_RESPONSE_OK` in order for the button to take over the file selected in the dialog.

Parameters

dialog	the widget to use as dialog.	[type Gtk.Dialog]

Returns

a new button widget.

Since: 2.6

---

gtk_file_chooser_button_get_title ()

```
const gchar *
gtk_file_chooser_button_get_title (GtkFileChooserButton *button);
```

Retrieves the title of the browse dialog used by `button` . The returned value should not be modified or freed.

Parameters

button	the button widget to examine.

Returns

a pointer to the browse dialog’s title.

Since: 2.6

---

gtk_file_chooser_button_set_title ()

```
void
gtk_file_chooser_button_set_title (GtkFileChooserButton *button,
                                   const gchar *title);
```

Modifies the `title` of the browse dialog used by `button` .

Parameters

button	the button widget to modify.
title	the new browse dialog title.

Since: 2.6

---

gtk_file_chooser_button_get_width_chars ()

```
gint
gtk_file_chooser_button_get_width_chars
                                (GtkFileChooserButton *button);
```

Retrieves the width in characters of the `button` widget's entry and/or label.

Parameters

button	the button widget to examine.

Returns

an integer width (in characters) that the button will use to size itself.

Since: 2.6

---

gtk_file_chooser_button_set_width_chars ()

```
void
gtk_file_chooser_button_set_width_chars
                                (GtkFileChooserButton *button,
                                 gint n_chars);
```

Sets the width (in characters) that `button` will use to `n_chars` .

Parameters

button	the button widget to examine.
n_chars	the new width, in characters.

Since: 2.6

---

gtk_file_chooser_button_get_focus_on_click ()

```
gboolean
gtk_file_chooser_button_get_focus_on_click
                                (GtkFileChooserButton *button);
```

`gtk_file_chooser_button_get_focus_on_click` has been deprecated since version 3.20 and should not be used in newly-written code.

Use `gtk_widget_get_focus_on_click()` instead

Returns whether the button grabs focus when it is clicked with the mouse. See `gtk_file_chooser_button_set_focus_on_click()`.

Parameters

button	a GtkFileChooserButton

Returns

`TRUE` if the button grabs focus when it is clicked with the mouse.

Since: 2.10

---

gtk_file_chooser_button_set_focus_on_click ()

```
void
gtk_file_chooser_button_set_focus_on_click
                                (GtkFileChooserButton *button,
                                 gboolean focus_on_click);
```

`gtk_file_chooser_button_set_focus_on_click` has been deprecated since version 3.20 and should not be used in newly-written code.

Use `gtk_widget_set_focus_on_click()` instead

Sets whether the button will grab focus when it is clicked with the mouse. Making mouse clicks not grab focus is useful in places like toolbars where you don’t want the keyboard focus removed from the main area of the application.

Parameters

button	a GtkFileChooserButton
focus_on_click	whether the button grabs focus when clicked with the mouse

Since: 2.10

Types and Values

struct GtkFileChooserButton

```
struct GtkFileChooserButton;
```

---

struct GtkFileChooserButtonClass

```
struct GtkFileChooserButtonClass {
  GtkBoxClass parent_class;
```

```
  void (* file_set) (GtkFileChooserButton *fc);
};
```

Members

`file_set` ()	Signal emitted when the user selects a file.

Property Details

The `“dialog”` property

`“dialog”` `GtkFileChooser *`

Instance of the GtkFileChooserDialog associated with the button.

Flags: Write / Construct Only

Since: 2.6

The `“title”` property

`“title”` `gchar *`

Title to put on the GtkFileChooserDialog associated with the button.

Flags: Read / Write

Default value: "Select a File"

Since: 2.6

The `“width-chars”` property

`“width-chars”` `gint`

The width of the entry and label inside the button, in characters.

Flags: Read / Write

Allowed values: >= -1

Default value: -1

Since: 2.6

Signal Details

The “file-set” signal

```
void
user_function (GtkFileChooserButton *widget,
               gpointer              user_data)
```

The ::file-set signal is emitted when the user selects a file.

Note that this signal is only emitted when the user changes the file.

Parameters

widget	the object which received the signal.
user_data	user data set when the signal handler was connected.

Flags: Run First

Since: 2.12

See Also

GtkFileChooserDialog

GtkFileChooserNative

GtkFileChooserNative — A native file chooser dialog, suitable for “File/Open” or “File/Save” commands

Functions

GtkFileChooserNative *	gtk_file_chooser_native_new ()
const char *	gtk_file_chooser_native_get_accept_label ()
void	gtk_file_chooser_native_set_accept_label ()
const char *	gtk_file_chooser_native_get_cancel_label ()
void	gtk_file_chooser_native_set_cancel_label ()

Includes

```
#include <gtk/gtk.h>
```

Description

GtkFileChooserNative is an abstraction of a dialog box suitable for use with “File/Open” or “File/Save as” commands. By default, this just uses a GtkFileChooserDialog to implement the actual dialog. However, on certain platforms, such as Windows and macOS, the native platform file chooser is used instead. When the application is running in a sandboxed environment without direct filesystem access (such as Flatpak), GtkFileChooserNative may call the proper APIs (portals) to let the user choose a file and make it available to the application.

While the API of GtkFileChooserNative closely mirrors GtkFileChooserDialog, the main difference is that there

is no access to any GtkWindow or GtkWidget for the dialog. This is required, as there may not be one in the case of a platform native dialog. Showing, hiding and running the dialog is handled by the GtkNativeDialog functions.

Typical usage

In the simplest of cases, you can the following code to use GtkFileChooserDialog to select a file for opening:

```
GtkFileChooserNative *native;
GtkFileChooserAction action =
GTK_FILE_CHOOSER_ACTION_OPEN;
gint res;

native = gtk_file_chooser_native_new ("Open
File",

parent_window,
                                      action,

"_Open",

"_Cancel");

res = gtk_native_dialog_run
(GTK_NATIVE_DIALOG (native));
if (res == GTK_RESPONSE_ACCEPT)
  {
    char *filename;
    GtkFileChooser *chooser =
GTK_FILE_CHOOSER (native);
    filename = gtk_file_chooser_get_filename
(chooser);
    open_file (filename);
    g_free (filename);
  }

g_object_unref (native);
```

To use a dialog for saving, you can use this:

```
GtkFileChooserNative *native;
GtkFileChooser *chooser;
GtkFileChooserAction action =
GTK_FILE_CHOOSER_ACTION_SAVE;
gint res;

native = gtk_file_chooser_native_new ("Save
File",

parent_window,
                                      action,

"_Save",

"_Cancel");
chooser = GTK_FILE_CHOOSER (native);

gtk_file_chooser_set_do_overwrite_confirmatio
n (chooser, TRUE);

if (user_edited_a_new_document)
  gtk_file_chooser_set_current_name (chooser,
```

```
24                                                  _("Untitled document"));
25                                                  else
26                                                    gtk_file_chooser_set_filename (chooser,
27
28                                                  existing_filename);
29
30                                                  res = gtk_native_dialog_run
31                                                  (GTK_NATIVE_DIALOG (native));
32                                                  if (res == GTK_RESPONSE_ACCEPT)
                                                      {
                                                        char *filename;

                                                        filename = gtk_file_chooser_get_filename
                                                    (chooser);
                                                        save_to_file (filename);
                                                        g_free (filename);
                                                      }

                                                    g_object_unref (native);
```

For more information on how to best set up a file dialog, see GtkFileChooserDialog.

Response Codes

GtkFileChooserNative inherits from GtkNativeDialog, which means it will return GTK_RESPONSE_ACCEPT if the user accepted, and GTK_RESPONSE_CANCEL if he pressed cancel. It can also return GTK_RESPONSE_DELETE_EVENT if the window was unexpectedly closed.

Differences from GtkFileChooserDialog

There are a few things in the GtkFileChooser API that are not possible to use with GtkFileChooserNative, as such use would prohibit the use of a native dialog.

There is no support for the signals that are emitted when the user navigates in the dialog, including:

- “current-folder-changed”
- “selection-changed”
- “file-activated”
- “confirm-overwrite”

You can also not use the methods that directly control user navigation:

- gtk_file_chooser_unselect_filename()
- gtk_file_chooser_select_all()
- gtk_file_chooser_unselect_all()

If you need any of the above you will have to use GtkFileChooserDialog directly.

No operations that change the the dialog work while the dialog is visible. Set all the properties that are required before showing the dialog.

Win32 details

On windows the IFileDialog implementation (added in Windows Vista) is used. It supports many of the features

that GtkFileChooserDialog does, but there are some things it does not handle:

- Extra widgets added with `gtk_file_chooser_set_extra_widget()`.
- Use of custom previews by connecting to "update-preview".
- Any GtkFileFilter added using a mimetype or custom filter.

If any of these features are used the regular GtkFileChooserDialog will be used in place of the native one.

Portal details

When the org.freedesktop.portal.FileChooser portal is available on the session bus, it is used to bring up an out-of-process file chooser. Depending on the kind of session the application is running in, this may or may not be a GTK+ file chooser. In this situation, the following things are not supported and will be silently ignored:

- Extra widgets added with `gtk_file_chooser_set_extra_widget()`.
- Use of custom previews by connecting to "update-preview".
- Any GtkFileFilter added with a custom filter.

macOS details

On macOS the NSSavePanel and NSOpenPanel classes are used to provide native file chooser dialogs. Some features provided by GtkFileChooserDialog are not supported:

- Extra widgets added with `gtk_file_chooser_set_extra_widget()`, unless the widget is an instance of GtkLabel, in which case the label text will be used to set the NSSavePanel message instance property.
- Use of custom previews by connecting to "update-preview".
- Any GtkFileFilter added with a custom filter.
- Shortcut folders.

Functions

gtk_file_chooser_native_new ()

```
GtkFileChooserNative *
gtk_file_chooser_native_new (const gchar *title,
                             GtkWindow *parent,
                             GtkFileChooserAction action,
                             const gchar *accept_label,
                             const gchar *cancel_label);
```

Creates a new GtkFileChooserNative.

Parameters

title	Title of the native, or `NULL`.	[allow-none]
parent	Transient parent of the native, or `NULL`.	[allow-none]

action	Open or save mode for the dialog	
accept_label	text to go in the accept button, or `NULL` for the default.	[allow-none]
cancel_label	text to go in the cancel button, or `NULL` for the default.	[allow-none]

Returns

a new GtkFileChooserNative

Since: 3.20

---

gtk_file_chooser_native_get_accept_label ()

```
const char *
gtk_file_chooser_native_get_accept_label
                                (GtkFileChooserNative *self);
```

Retrieves the custom label text for the accept button.

Parameters

self	a GtFileChooserNative

Returns

The custom label, or `NULL` for the default. This string is owned by GTK+ and should not be modified or freed.

[nullable]

Since: 3.20

---

gtk_file_chooser_native_set_accept_label ()

```
void
gtk_file_chooser_native_set_accept_label
                                (GtkFileChooserNative *self,
                                 const char *accept_label);
```

Sets the custom label text for the accept button.

If characters in `label` are preceded by an underscore, they are underlined. If you need a literal underscore character in a label, use “__” (two underscores). The first underlined character represents a keyboard accelerator called a mnemonic. Pressing Alt and that key activates the button.

Parameters

self	a GtFileChooserNative	
accept_label	custom label or `NULL` for the default.	[allow-none]

Since: 3.20

gtk_file_chooser_native_get_cancel_label ()

```
const char *
gtk_file_chooser_native_get_cancel_label
                                (GtkFileChooserNative *self);
```

Retrieves the custom label text for the cancel button.

Parameters

self	a GtFileChooserNative

Returns

The custom label, or `NULL` for the default. This string is owned by GTK+ and should not be modified or freed.

[nullable]

Since: 3.20

---

gtk_file_chooser_native_set_cancel_label ()

```
void
gtk_file_chooser_native_set_cancel_label
                                (GtkFileChooserNative *self,
                                 const char *cancel_label);
```

Sets the custom label text for the cancel button.

If characters in `label` are preceded by an underscore, they are underlined. If you need a literal underscore character in a label, use "__" (two underscores). The first underlined character represents a keyboard accelerator called a mnemonic. Pressing Alt and that key activates the button.

Parameters

self	a GtFileChooserNative
cancel_label	custom label or `NULL` for the default. [allow-none]

Since: 3.20

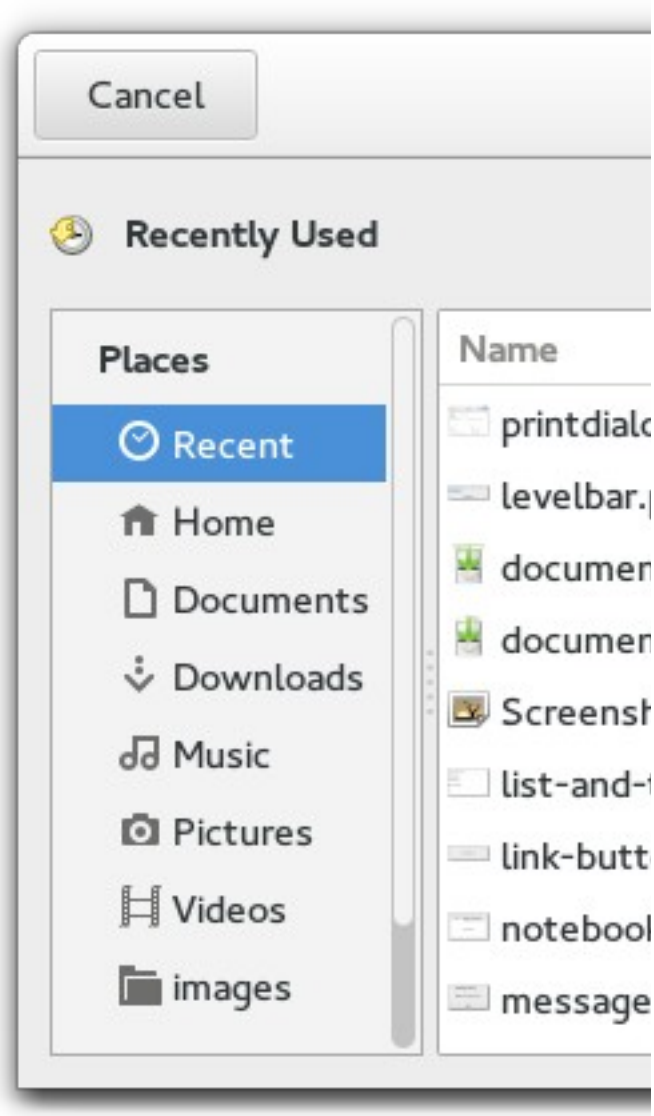
Types and Values

See Also

GtkFileChooser, GtkNativeDialog, GtkFileChooserDialog

GtkFileChooserDialog

GtkFileChooserDialog — A file chooser dialog, suitable for “File/Open” or “File/Save” commands



Functions

[GtkWidget](#) *	[gtk_file_chooser_dialog_new](#) ()

Types and Values

struct	[GtkFileChooserDialog](#)

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkContainer
            └── GtkBin
                └── GtkWindow
                    └── GtkDialog
```

```
└── GtkFileChooserDialog
```

Implemented Interfaces

GtkFileChooserDialog implements AtkImplementorIface, GtkBuildable and GtkFileChooser.

Includes

```
#include <gtk/gtk.h>
```

Description

GtkFileChooserDialog is a dialog box suitable for use with “File/Open” or “File/Save as” commands. This widget works by putting a GtkFileChooserWidget inside a GtkDialog. It exposes the GtkFileChooser interface, so you can use all of the GtkFileChooser functions on the file chooser dialog as well as those for GtkDialog.

Note that GtkFileChooserDialog does not have any methods of its own. Instead, you should use the functions that work on a GtkFileChooser.

If you want to integrate well with the platform you should use the GtkFileChooserNative API, which will use a platform-specific dialog if available and fall back to GtkFileChooserDialog otherwise.

Typical usage

In the simplest of cases, you can the following code to use GtkFileChooserDialog to select a file for opening:

```
GtkWidget *dialog;
GtkFileChooserAction action =
GTK_FILE_CHOOSER_ACTION_OPEN;
gint res;

dialog = gtk_file_chooser_dialog_new ("Open
File",

parent_window,
                                      action,

_("_Cancel"),

GTK_RESPONSE_CANCEL,

_("_Open"),

GTK_RESPONSE_ACCEPT,
                                      NULL);

res = gtk_dialog_run (GTK_DIALOG (dialog));
if (res == GTK_RESPONSE_ACCEPT)
  {
    char *filename;
    GtkFileChooser *chooser =
GTK_FILE_CHOOSER (dialog);
    filename = gtk_file_chooser_get_filename
(chooser);
    open_file (filename);
    g_free (filename);
```

```
  }

gtk_widget_destroy (dialog);
```

To use a dialog for saving, you can use this:

```
GtkWidget *dialog;
GtkFileChooser *chooser;
GtkFileChooserAction action =
GTK_FILE_CHOOSER_ACTION_SAVE;
gint res;

dialog = gtk_file_chooser_dialog_new ("Save
File",

parent_window,
                                      action,

_("_Cancel"),

GTK_RESPONSE_CANCEL,

_("_Save"),

GTK_RESPONSE_ACCEPT,
                                      NULL);
chooser = GTK_FILE_CHOOSER (dialog);

gtk_file_chooser_set_do_overwrite_confirmatio
n (chooser, TRUE);

if (user_edited_a_new_document)
  gtk_file_chooser_set_current_name (chooser,

_("Untitled document"));
else
  gtk_file_chooser_set_filename (chooser,

existing_filename);

res = gtk_dialog_run (GTK_DIALOG (dialog));
if (res == GTK_RESPONSE_ACCEPT)
  {
    char *filename;

    filename = gtk_file_chooser_get_filename
(chooser);
    save_to_file (filename);
    g_free (filename);
  }

gtk_widget_destroy (dialog);
```

Setting up a file chooser dialog

There are various cases in which you may need to use a GtkFileChooserDialog:

- To select a file for opening. Use GTK_FILE_CHOOSER_ACTION_OPEN.
- To save a file for the first time. Use GTK_FILE_CHOOSER_ACTION_SAVE, and suggest a name such as “Untitled” with `gtk_file_chooser_set_current_name()`.
- To save a file under a different name. Use GTK_FILE_CHOOSER_ACTION_SAVE, and set the existing

filename with gtk_file_chooser_set_filename().

- To choose a folder instead of a file. Use GTK_FILE_CHOOSER_ACTION_SELECT_FOLDER.

Note that old versions of the file chooser's documentation suggested using gtk_file_chooser_set_current_folder() in various situations, with the intention of letting the application suggest a reasonable default folder. This is no longer considered to be a good policy, as now the file chooser is able to make good suggestions on its own. In general, you should only cause the file chooser to show a specific folder when it is appropriate to use gtk_file_chooser_set_filename(), i.e. when you are doing a Save As command and you already have a file saved somewhere.

Response Codes

GtkFileChooserDialog inherits from GtkDialog, so buttons that go in its action area have response codes such as GTK_RESPONSE_ACCEPT and GTK_RESPONSE_CANCEL. For example, you could call gtk_file_chooser_dialog_new() as follows:

```
GtkWidget *dialog;
GtkFileChooserAction action =
GTK_FILE_CHOOSER_ACTION_OPEN;

dialog = gtk_file_chooser_dialog_new ("Open
File",

parent_window,
                                      action,

_("_Cancel"),

GTK_RESPONSE_CANCEL,

_("_Open"),

GTK_RESPONSE_ACCEPT,
                                      NULL);
```

This will create buttons for "Cancel" and "Open" that use stock response identifiers from GtkResponseType. For most dialog boxes you can use your own custom response codes rather than the ones in GtkResponseType, but GtkFileChooserDialog assumes that its "accept"-type action, e.g. an "Open" or "Save" button, will have one of the following response codes:

- GTK_RESPONSE_ACCEPT
- GTK_RESPONSE_OK
- GTK_RESPONSE_YES
- GTK_RESPONSE_APPLY

This is because GtkFileChooserDialog must intercept responses and switch to folders if appropriate, rather than letting the dialog terminate — the implementation uses these known response codes to know which responses can be blocked if appropriate.

To summarize, make sure you use a stock response code when you use GtkFileChooserDialog to ensure proper operation.

Functions

gtk_file_chooser_dialog_new ()

```
GtkWidget *
gtk_file_chooser_dialog_new (const gchar *title,
                             GtkWindow *parent,
                             GtkFileChooserAction action,
                             const gchar *first_button_text,
                             ...);
```

Creates a new GtkFileChooserDialog. This function is analogous to `gtk_dialog_new_with_buttons()`.

Parameters

title	Title of the dialog, or `NULL`.	[allow-none]
parent	Transient parent of the dialog, or `NULL`.	[allow-none]
action	Open or save mode for the dialog	
first_button_text	stock ID or text to go in the first button, or `NULL`.	[allow-none]
...	response ID for the first button, then additional (button, id) pairs, ending with `NULL`	

Returns

a new GtkFileChooserDialog

Since: 2.4

Types and Values

struct GtkFileChooserDialog

```
struct GtkFileChooserDialog;
```

See Also

GtkFileChooser, GtkDialog, GtkFileChooserNative

GtkFileChooserWidget

GtkFileChooserWidget — A file chooser widget

Functions

GtkWidget *	gtk_file_chooser_widget_new ()

Properties

gboolean	search-mode	Read / Write
gchar *	subtitle	Read

Signals

void	desktop-folder	Action
void	down-folder	Action
void	home-folder	Action
void	location-popup	Action
void	location-popup-on-paste	Action
void	location-toggle-popup	Action
void	places-shortcut	Action
void	quick-bookmark	Action
void	recent-shortcut	Action
void	search-shortcut	Action
void	show-hidden	Action
void	up-folder	Action

Types and Values

struct	GtkFileChooserWidget
struct	GtkFileChooserWidgetClass

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkContainer
            └── GtkBox
                └── GtkFileChooserWidget
```

Implemented Interfaces

GtkFileChooserWidget implements AtkImplementorIface, GtkBuildable, GtkOrientable, GtkFileChooser and GtkFileChooserEmbed.

Includes

```
#include <gtk/gtk.h>
```

Description

GtkFileChooserWidget is a widget for choosing files. It exposes the GtkFileChooser interface, and you should use the methods of this interface to interact with the widget.

CSS nodes

GtkFileChooserWidget has a single CSS node with name filechooser.

Functions

gtk_file_chooser_widget_new ()

```
GtkWidget *
gtk_file_chooser_widget_new (GtkFileChooserAction action);
```

Creates a new GtkFileChooserWidget. This is a file chooser widget that can be embedded in custom windows, and it is the same widget that is used by GtkFileChooserDialog.

Parameters

action	Open or save mode for the widget

Returns

a new GtkFileChooserWidget

Since: 2.4

Types and Values

struct GtkFileChooserWidget

```
struct GtkFileChooserWidget;
```

---

struct GtkFileChooserWidgetClass

```
struct GtkFileChooserWidgetClass {
  GtkBoxClass parent_class;
};
```

Members

Property Details

The “search-mode” property

“search-mode”	gboolean

Search mode.

Flags: Read / Write

Default value: FALSE

The “subtitle” property

```
  “subtitle”                 gchar *
```

Subtitle.

Flags: Read

Default value: ""

Signal Details

The “desktop-folder” signal

```
void
user_function (GtkFileChooserWidget *widget,
               gpointer              user_data)
```

The ::desktop-folder signal is a keybinding signal which gets emitted when the user asks for it.

This is used to make the file chooser show the user's Desktop folder in the file list.

The default binding for this signal is `Alt + D`.

Parameters

widget	the object which received the signal
user_data	user data set when the signal handler was connected.

Flags: Action

The “down-folder” signal

```
void
user_function (GtkFileChooserWidget *widget,
               gpointer              user_data)
```

The ::down-folder signal is a keybinding signal which gets emitted when the user asks for it.

This is used to make the file chooser go to a child of the current folder in the file hierarchy. The subfolder that will be used is displayed in the path bar widget of the file chooser. For example, if the path bar is showing "/foo/bar/baz", with bar currently displayed, then this will cause the file chooser to switch to the "baz" subfolder.

The default binding for this signal is `Alt + Down`.

Parameters

widget	the object which received the signal
user_data	user data set when the signal handler was connected.

Flags: Action

The `"home-folder"` signal

```
void
user_function (GtkFileChooserWidget *widget,
               gpointer              user_data)
```

The ::home-folder signal is a keybinding signal which gets emitted when the user asks for it.

This is used to make the file chooser show the user's home folder in the file list.

The default binding for this signal is `Alt + Home`.

Parameters

widget	the object which received the signal
user_data	user data set when the signal handler was connected.

Flags: Action

The `"location-popup"` signal

```
void
user_function (GtkFileChooserWidget *widget,
               gchar                *path,
               gpointer              user_data)
```

The ::location-popup signal is a keybinding signal which gets emitted when the user asks for it.

This is used to make the file chooser show a "Location" prompt which the user can use to manually type the name of the file he wishes to select.

The default bindings for this signal are `Control + L` with a `path` string of "" (the empty string). It is also bound to `/` with a `path` string of "`/`" (a slash): this lets you type `/` and immediately type a path name. On Unix systems, this is bound to ~ (tilde) with a `path` string of "~" itself for access to home directories.

Parameters

widget	the object which received the signal
path	a string that gets put in the text entry for the file name
user_data	user data set when the signal handler was connected.

Flags: Action

The “location-popup-on-paste” signal

```
void
user_function (GtkFileChooserWidget *widget,
               gpointer              user_data)
```

The ::location-popup-on-paste signal is a keybinding signal which gets emitted when the user asks for it.

This is used to make the file chooser show a "Location" prompt when the user pastes into a GtkFileChooserWidget.

The default binding for this signal is `Control + V`.

Parameters

widget	the object which received the signal
user_data	user data set when the signal handler was connected.

Flags: Action

The “location-toggle-popup” signal

```
void
user_function (GtkFileChooserWidget *widget,
               gpointer              user_data)
```

The ::location-toggle-popup signal is a keybinding signal which gets emitted when the user asks for it.

This is used to toggle the visibility of a "Location" prompt which the user can use to manually type the name of the file he wishes to select.

The default binding for this signal is `Control + L`.

Parameters

widget	the object which received the signal
user_data	user data set when the signal handler was connected.

Flags: Action

The “places-shortcut” signal

```
void
user_function (GtkFileChooserWidget *widget,
               gpointer              user_data)
```

The ::places-shortcut signal is a keybinding signal which gets emitted when the user asks for it.

This is used to move the focus to the places sidebar.

The default binding for this signal is `Alt + P`.

Parameters

widget	the object which received the signal
user_data	user data set when the signal handler was connected.

Flags: Action

The `"quick-bookmark"` signal

```
void
user_function (GtkFileChooserWidget *widget,
               gint                  bookmark_index,
               gpointer              user_data)
```

The ::quick-bookmark signal is a keybinding signal which gets emitted when the user asks for it.

This is used to make the file chooser switch to the bookmark specified in the `bookmark_index` parameter. For example, if you have three bookmarks, you can pass 0, 1, 2 to this signal to switch to each of them, respectively.

The default binding for this signal is `Alt + 1`, `Alt + 2`, etc. until `Alt + 0`. Note that in the default binding, that `Alt + 1` is actually defined to switch to the bookmark at index 0, and so on successively; `Alt + 0` is defined to switch to the bookmark at index 10.

Parameters

widget	the object which received the signal
bookmark_index	the number of the bookmark to switch to
user_data	user data set when the signal handler was connected.

Flags: Action

The `"recent-shortcut"` signal

```
void
user_function (GtkFileChooserWidget *widget,
               gpointer              user_data)
```

The ::recent-shortcut signal is a keybinding signal which gets emitted when the user asks for it.

This is used to make the file chooser show the Recent location.

The default binding for this signal is `Alt + R`.

Parameters

widget	the object which received the signal
user_data	user data set when the signal handler was connected.

Flags: Action

The “search-shortcut” signal

```
void
user_function (GtkFileChooserWidget *widget,
               gpointer              user_data)
```

The ::search-shortcut signal is a keybinding signal which gets emitted when the user asks for it.

This is used to make the file chooser show the search entry.

The default binding for this signal is `Alt + S`.

Parameters

widget	the object which received the signal
user_data	user data set when the signal handler was connected.

Flags: Action

The “show-hidden” signal

```
void
user_function (GtkFileChooserWidget *widget,
               gpointer              user_data)
```

The ::show-hidden signal is a keybinding signal which gets emitted when the user asks for it.

This is used to make the file chooser display hidden files.

The default binding for this signal is `Control + H`.

Parameters

widget	the object which received the signal
user_data	user data set when the signal handler was connected.

Flags: Action

The “up-folder” signal

```
void
user_function (GtkFileChooserWidget *widget,
               gpointer              user_data)
```

The ::up-folder signal is a keybinding signal which gets emitted when the user asks for it.

This is used to make the file chooser go to the parent of the current folder in the file hierarchy.

The default binding for this signal is `Alt + Up`.

Parameters

widget	the object which received the signal
user_data	user data set when the signal handler was connected.

Flags: Action

See Also

GtkFileChooserDialog

GtkFileFilter

GtkFileFilter — A filter for selecting a file subset

Functions

gboolean	(*GtkFileFilterFunc) ()
GtkFileFilter *	gtk_file_filter_new ()
void	gtk_file_filter_set_name ()
const gchar *	gtk_file_filter_get_name ()
void	gtk_file_filter_add_mime_type ()
void	gtk_file_filter_add_pattern ()
void	gtk_file_filter_add_pixbuf_formats ()
void	gtk_file_filter_add_custom ()
GtkFileFilterFlags	gtk_file_filter_get_needed ()
gboolean	gtk_file_filter_filter ()
GtkFileFilter *	gtk_file_filter_new_from_gvariant ()
GVariant *	gtk_file_filter_to_gvariant ()

Types and Values

	GtkFileFilter
struct	GtkFileFilterInfo
enum	GtkFileFilterFlags

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkFileFilter
```

Implemented Interfaces

GtkFileFilter implements GtkBuildable.

Includes

```
#include <gtk/gtk.h>
```

Description

A GtkFileFilter can be used to restrict the files being shown in a GtkFileChooser. Files can be filtered based on their name (with `gtk_file_filter_add_pattern()`), on their mime type (with `gtk_file_filter_add_mime_type()`), or by a custom filter function (with `gtk_file_filter_add_custom()`).

Filtering by mime types handles aliasing and subclassing of mime types; e.g. a filter for text/plain also matches a file with mime type application/rtf, since application/rtf is a subclass of text/plain. Note that GtkFileFilter allows wildcards for the subtype of a mime type, so you can e.g. filter for image/*.

Normally, filters are used by adding them to a GtkFileChooser, see `gtk_file_chooser_add_filter()`, but it is also possible to manually use a filter on a file with `gtk_file_filter_filter()`.

GtkFileFilter as GtkBuildable

The GtkFileFilter implementation of the GtkBuildable interface supports adding rules using the <mime-types>, <patterns> and <applications> elements and listing the rules within. Specifying a <mime-type> or <pattern> has the same effect as as calling `gtk_file_filter_add_mime_type()` or `gtk_file_filter_add_pattern()`.

An example of a UI definition fragment specifying GtkFileFilter rules:

```
<object class="GtkFileFilter">
  <mime-types>
    <mime-type>text/plain</mime-type>
    <mime-type>image/ *</mime-type>
  </mime-types>
  <patterns>
    <pattern>*.txt</pattern>
    <pattern>*.png</pattern>
  </patterns>
</object>
```

Functions

GtkFileFilterFunc ()

```
gboolean
(*GtkFileFilterFunc) (const GtkFileFilterInfo *filter_info,
                      gpointer data);
```

The type of function that is used with custom filters, see `gtk_file_filter_add_custom()`.

Parameters

filter_info	a GtkFileFilterInfo that is filled according to the `needed` flags

	passed to `gtk_file_filter_add_custom()`	
data	user data passed to `gtk_file_filter_add_custom()`.	[closure]

Returns

`TRUE` if the file should be displayed

---

gtk_file_filter_new ()

```
GtkFileFilter *
gtk_file_filter_new (void);
```

Creates a new GtkFileFilter with no rules added to it. Such a filter doesn't accept any files, so is not particularly useful until you add rules with `gtk_file_filter_add_mime_type()`, `gtk_file_filter_add_pattern()`, or `gtk_file_filter_add_custom()`. To create a filter that accepts any file, use:

```
1 GtkFileFilter *filter = gtk_file_filter_new ();
2 gtk_file_filter_add_pattern (filter, "*");
```

Returns

a new GtkFileFilter

Since: 2.4

---

gtk_file_filter_set_name ()

```
void
gtk_file_filter_set_name (GtkFileFilter *filter,
                          const gchar *name);
```

Sets the human-readable name of the filter; this is the string that will be displayed in the file selector user interface if there is a selectable list of filters.

Parameters

filter	a GtkFileFilter	
name	the human-readable-name for the filter, or `NULL` to remove any existing name.	[allow-none]

Since: 2.4

---

gtk_file_filter_get_name ()

```
const gchar *
gtk_file_filter_get_name (GtkFileFilter *filter);
```

Gets the human-readable name for the filter. See `gtk_file_filter_set_name()`.

Parameters

filter	a GtkFileFilter

Returns

The human-readable name of the filter, or `NULL`. This value is owned by GTK+ and must not be modified or freed.

[nullable]

Since: 2.4

---

gtk_file_filter_add_mime_type ()

```
void
gtk_file_filter_add_mime_type (GtkFileFilter *filter,
                               const gchar *mime_type);
```

Adds a rule allowing a given mime type to `filter` .

Parameters

filter	A GtkFileFilter
mime_type	name of a MIME type

Since: 2.4

---

gtk_file_filter_add_pattern ()

```
void
gtk_file_filter_add_pattern (GtkFileFilter *filter,
                             const gchar *pattern);
```

Adds a rule allowing a shell style glob to a filter.

Parameters

filter	a GtkFileFilter
pattern	a shell style glob

Since: 2.4

---

gtk_file_filter_add_pixbuf_formats ()

```
void
```

```
gtk_file_filter_add_pixbuf_formats (GtkFileFilter *filter);
```
Adds a rule allowing image files in the formats supported by GdkPixbuf.

Parameters

filter	a GtkFileFilter

Since: 2.6

---

gtk_file_filter_add_custom ()

```
void
gtk_file_filter_add_custom (GtkFileFilter *filter,
                            GtkFileFilterFlags needed,
                            GtkFileFilterFunc func,
                            gpointer data,
                            GDestroyNotify notify);
```
Adds rule to a filter that allows files based on a custom callback function. The bitfield `needed` which is passed in provides information about what sorts of information that the filter function needs; this allows GTK+ to avoid retrieving expensive information when it isn't needed by the filter.

Parameters

filter	a GtkFileFilter
needed	bitfield of flags indicating the information that the custom filter function needs.
func	callback function; if the function returns `TRUE`, then the file will be displayed.
data	data to pass to `func`
notify	function to call to free `data` when it is no longer needed.

Since: 2.4

---

gtk_file_filter_get_needed ()

```
GtkFileFilterFlags
gtk_file_filter_get_needed (GtkFileFilter *filter);
```
Gets the fields that need to be filled in for the GtkFileFilterInfo passed to `gtk_file_filter_filter()`

This function will not typically be used by applications; it is intended principally for use in the implementation of GtkFileChooser.

Parameters

filter	a GtkFileFilter

Returns

bitfield of flags indicating needed fields when calling `gtk_file_filter_filter()`

Since: 2.4

---

gtk_file_filter_filter ()

```
gboolean
gtk_file_filter_filter (GtkFileFilter *filter,
                        const GtkFileFilterInfo *filter_info);
```

Tests whether a file should be displayed according to `filter` . The GtkFileFilterInfo `filter_info` should include the fields returned from `gtk_file_filter_get_needed()`.

This function will not typically be used by applications; it is intended principally for use in the implementation of GtkFileChooser.

Parameters

filter	a GtkFileFilter
filter_info	a GtkFileFilterInfo containing information about a file.

Returns

`TRUE` if the file should be displayed

Since: 2.4

---

gtk_file_filter_new_from_gvariant ()

```
GtkFileFilter *
gtk_file_filter_new_from_gvariant (GVariant *variant);
```

Deserialize a file filter from an a{sv} variant in the format produced by `gtk_file_filter_to_gvariant()`.

Parameters

variant	an a{sv} GVariant

Returns

a new GtkFileFilter object.

[transfer full]

Since: 3.22

---

gtk_file_filter_to_gvariant ()

```
GVariant *
gtk_file_filter_to_gvariant (GtkFileFilter *filter);
```

Serialize a file filter to an a{sv} variant.

Parameters

filter	a GtkFileFilter

Returns

a new, floating, GVariant.

[transfer none]

Since: 3.22

Types and Values

GtkFileFilter

```
typedef struct _GtkFileFilter GtkFileFilter;
```

---

struct GtkFileFilterInfo

```
struct GtkFileFilterInfo {
  GtkFileFilterFlags contains;

  const gchar *filename;
  const gchar *uri;
  const gchar *display_name;
  const gchar *mime_type;
};
```

A GtkFileFilterInfo is used to pass information about the tested file to `gtk_file_filter_filter()`.

Members

GtkFileFilterFlags `contains`;	Flags indicating which of the following fields need are filled
const gchar *`filename`;	the filename of the file being tested
const gchar *`uri`;	the URI for the file being tested
const gchar *`display_name`;	the string that will be used to display the file in the file chooser
const gchar *`mime_type`;	the mime type of the file

---

enum GtkFileFilterFlags

These flags indicate what parts of a GtkFileFilterInfo struct are filled or need to be filled.

Members

GTK_FILE_FILTER_FILENAME	the filename of the file being tested
GTK_FILE_FILTER_URI	the URI for the file being tested
GTK_FILE_FILTER_DISPLAY_NAME	the string that will be used to display the file in the file chooser
GTK_FILE_FILTER_MIME_TYPE	the mime type of the file

See Also

GtkFileChooser

GtkFontChooser

GtkFontChooser — Interface implemented by widgets displaying fonts

Functions

PangoFontFamily *	gtk_font_chooser_get_font_family ()
PangoFontFace *	gtk_font_chooser_get_font_face ()
gint	gtk_font_chooser_get_font_size ()
gchar *	gtk_font_chooser_get_font ()
void	gtk_font_chooser_set_font ()
PangoFontDescription *	gtk_font_chooser_get_font_desc ()
void	gtk_font_chooser_set_font_desc ()
gchar *	gtk_font_chooser_get_preview_text ()
void	gtk_font_chooser_set_preview_text ()
gboolean	gtk_font_chooser_get_show_preview_entry ()
void	gtk_font_chooser_set_show_preview_entry ()
gboolean	(*GtkFontFilterFunc) ()
void	gtk_font_chooser_set_filter_func ()
void	gtk_font_chooser_set_font_map ()
PangoFontMap *	gtk_font_chooser_get_font_map ()
void	gtk_font_chooser_set_level ()
GtkFontChooserLevel	gtk_font_chooser_get_level ()
char *	gtk_font_chooser_get_font_features ()
void	gtk_font_chooser_set_language ()
char *	gtk_font_chooser_get_language ()

Properties

gchar *	font	Read / Write

PangoFontDescription *	font-desc	Read / Write
gchar *	font-features	Read
gchar *	language	Read / Write
GtkFontChooserLevel	level	Read / Write
gchar *	preview-text	Read / Write
gboolean	show-preview-entry	Read / Write

Signals

void	font-activated	Run First

Types and Values

GtkFontChooser

Object Hierarchy

```
GInterface
└── GtkFontChooser
```

Prerequisites

GtkFontChooser requires GObject.

Known Implementations

GtkFontChooser is implemented by GtkFontButton, GtkFontChooserDialog and GtkFontChooserWidget.

Includes

```
#include <gtk/gtk.h>
```

Description

GtkFontChooser is an interface that can be implemented by widgets displaying the list of fonts. In GTK+, the main objects that implement this interface are GtkFontChooserWidget, GtkFontChooserDialog and GtkFontButton. The GtkFontChooser interface has been introducted in GTK+ 3.2.

Functions

gtk_font_chooser_get_font_family ()

```
PangoFontFamily *
gtk_font_chooser_get_font_family (GtkFontChooser *fontchooser);
```

Gets the PangoFontFamily representing the selected font family. Font families are a collection of font faces.

If the selected font is not installed, returns `NULL`.

Parameters

fontchooser	a GtkFontChooser

Returns

A PangoFontFamily representing the selected font family, or `NULL`. The returned object is owned by `fontchooser` and must not be modified or freed.

[nullable][transfer none]

Since: 3.2

---

gtk_font_chooser_get_font_face ()

```
PangoFontFace *
gtk_font_chooser_get_font_face (GtkFontChooser *fontchooser);
```

Gets the PangoFontFace representing the selected font group details (i.e. family, slant, weight, width, etc).

If the selected font is not installed, returns `NULL`.

Parameters

fontchooser	a GtkFontChooser

Returns

A PangoFontFace representing the selected font group details, or `NULL`. The returned object is owned by `fontchooser` and must not be modified or freed.

[nullable][transfer none]

Since: 3.2

---

gtk_font_chooser_get_font_size ()

```
gint
gtk_font_chooser_get_font_size (GtkFontChooser *fontchooser);
```

The selected font size.

Parameters

fontchooser	a GtkFontChooser

Returns

A n integer representing the selected font size, or -1 if no font size is selected.

Since: 3.2

---

gtk_font_chooser_get_font ()

```
gchar *
gtk_font_chooser_get_font (GtkFontChooser *fontchooser);
```

Gets the currently-selected font name.

Note that this can be a different string than what you set with `gtk_font_chooser_set_font()`, as the font chooser widget may normalize font names and thus return a string with a different structure. For example, “Helvetica Italic Bold 12” could be normalized to “Helvetica Bold Italic 12”.

Use `pango_font_description_equal()` if you want to compare two font descriptions.

Parameters

fontchooser	a GtkFontChooser

Returns

A string with the name of the current font, or `NULL` if no font is selected. You must free this string with `g_free()`.

[nullable][transfer full]

Since: 3.2

---

gtk_font_chooser_set_font ()

```
void
gtk_font_chooser_set_font (GtkFontChooser *fontchooser,
                           const gchar *fontname);
```

Sets the currently-selected font.

Parameters

fontchooser	a GtkFontChooser
fontname	a font name like “Helvetica 12” or “Times Bold 18”

Since: 3.2

---

gtk_font_chooser_get_font_desc ()

```
PangoFontDescription *
gtk_font_chooser_get_font_desc (GtkFontChooser *fontchooser);
```

Gets the currently-selected font.

Note that this can be a different string than what you set with `gtk_font_chooser_set_font()`, as the font chooser widget may normalize font names and thus return a string with a different structure. For example, “Helvetica Italic Bold 12” could be normalized to “Helvetica Bold Italic 12”.

Use `pango_font_description_equal()` if you want to compare two font descriptions.

Parameters

fontchooser	a GtkFontChooser

Returns

A PangoFontDescription for the current font, or `NULL` if no font is selected.

[nullable][transfer full]

Since: 3.2

---

gtk_font_chooser_set_font_desc ()

```
void
gtk_font_chooser_set_font_desc (GtkFontChooser *fontchooser,
                                const PangoFontDescription *font_desc);
```

Sets the currently-selected font from `font_desc`.

Parameters

fontchooser	a GtkFontChooser
font_desc	a PangoFontDescription

Since: 3.2

---

gtk_font_chooser_get_preview_text ()

```
gchar *
gtk_font_chooser_get_preview_text (GtkFontChooser *fontchooser);
```

Gets the text displayed in the preview area.

Parameters

fontchooser	a GtkFontChooser

Returns

the text displayed in the preview area.

[transfer full]

Since: 3.2

---

gtk_font_chooser_set_preview_text ()

```
void
gtk_font_chooser_set_preview_text (GtkFontChooser *fontchooser,
                                   const gchar *text);
```

Sets the text displayed in the preview area. The `text` is used to show how the selected font looks.

Parameters

fontchooser	a GtkFontChooser	
text	the text to display in the preview area.	[transfer none]

Since: 3.2

---

gtk_font_chooser_get_show_preview_entry ()

```
gboolean
gtk_font_chooser_get_show_preview_entry
                                (GtkFontChooser *fontchooser);
```

Returns whether the preview entry is shown or not.

Parameters

fontchooser	a GtkFontChooser

Returns

`TRUE` if the preview entry is shown or `FALSE` if it is hidden.

Since: 3.2

---

gtk_font_chooser_set_show_preview_entry ()

```
void
gtk_font_chooser_set_show_preview_entry
                                (GtkFontChooser *fontchooser,
                                 gboolean show_preview_entry);
```

Shows or hides the editable preview entry.

Parameters

fontchooser	a GtkFontChooser	
show_preview_entry	whether to show the editable preview entry or not	

Since: 3.2

GtkFontFilterFunc ()

```
gboolean
(*GtkFontFilterFunc) (const PangoFontFamily *family,
                      const PangoFontFace *face,
                      gpointer data);
```

The type of function that is used for deciding what fonts get shown in a GtkFontChooser. See `gtk_font_chooser_set_filter_func()`.

Parameters

family	a PangoFontFamily	
face	a PangoFontFace belonging to `family`	
data	user data passed to `gtk_font_chooser_set_filter_func()`.	[closure]

Returns

`TRUE` if the font should be displayed

---

gtk_font_chooser_set_filter_func ()

```
void
gtk_font_chooser_set_filter_func (GtkFontChooser *fontchooser,
                                  GtkFontFilterFunc filter,
                                  gpointer user_data,
                                  GDestroyNotify destroy);
```

Adds a filter function that decides which fonts to display in the font chooser.

Parameters

fontchooser	a GtkFontChooser	
filter	a GtkFontFilterFunc, or `NULL`.	[allow-none]
user_data	data to pass to `filter`	
destroy	function to call to free `data` when it is no longer needed	

Since: 3.2

gtk_font_chooser_set_font_map ()

```
void
gtk_font_chooser_set_font_map (GtkFontChooser *fontchooser,
                               PangoFontMap *fontmap);
```

Sets a custom font map to use for this font chooser widget. A custom font map can be used to present application-specific fonts instead of or in addition to the normal system fonts.

```
FcConfig *config;
PangoFontMap *fontmap;

config = FcInitLoadConfigAndFonts ();
FcConfigAppFontAddFile (config,
my_app_font_file);

fontmap =
pango_cairo_font_map_new_for_font_type
(CAIRO_FONT_TYPE_FT);
pango_fc_font_map_set_config
(PANGO_FC_FONT_MAP (fontmap), config);

gtk_font_chooser_set_font_map (font_chooser,
fontmap);
```

Note that other GTK+ widgets will only be able to use the application-specific font if it is present in the font map they use:

```
context = gtk_widget_get_pango_context
(label);
pango_context_set_font_map (context,
fontmap);
```

Parameters

fontchooser	a GtkFontChooser	
fontmap	a PangoFontMap.	[allow-none]

Since: 3.18

---

gtk_font_chooser_get_font_map ()

```
PangoFontMap *
gtk_font_chooser_get_font_map (GtkFontChooser *fontchooser);
```

Gets the custom font map of this font chooser widget, or `NULL` if it does not have one.

Parameters

fontchooser	a GtkFontChooser

Returns

a PangoFontMap, or `NULL`.

[nullable][transfer full]

Since: 3.18

---

gtk_font_chooser_set_level ()

```
void
gtk_font_chooser_set_level (GtkFontChooser *fontchooser,
                            GtkFontChooserLevel level);
```

Sets the desired level of granularity for selecting fonts.

Parameters

fontchooser	a GtkFontChooser
level	the desired level of granularity

Since: 3.24

---

gtk_font_chooser_get_level ()

```
GtkFontChooserLevel
gtk_font_chooser_get_level (GtkFontChooser *fontchooser);
```

Returns the current level of granularity for selecting fonts.

Parameters

fontchooser	a GtkFontChooser

Returns

the current granularity level

Since: 3.24

---

gtk_font_chooser_get_font_features ()

```
char *
gtk_font_chooser_get_font_features (GtkFontChooser *fontchooser);
```

Gets the currently-selected font features.

Parameters

fontchooser	a GtkFontChooser

Returns

the currently selected font features

Since: 3.24

---

gtk_font_chooser_set_language ()

```
void
gtk_font_chooser_set_language (GtkFontChooser *fontchooser,
                               const char *language);
```

Sets the language to use for font features.

Parameters

fontchooser	a GtkFontChooser
language	a language

Since: 3.24

---

gtk_font_chooser_get_language ()

```
char *
gtk_font_chooser_get_language (GtkFontChooser *fontchooser);
```

Gets the language that is used for font features.

Parameters

fontchooser	a GtkFontChooser

Returns

the currently selected language

Since: 3.24

Types and Values

GtkFontChooser

```
typedef struct _GtkFontChooser GtkFontChooser;
```

Property Details

The “font” property

“font” gchar *

The font description as a string, e.g. "Sans Italic 12".

Flags: Read / Write

Default value: "Sans 10"

The “font-desc” property

“font-desc” PangoFontDescription *

The font description as a PangoFontDescription.

Flags: Read / Write

The “font-features” property

“font-features” gchar *

The selected font features, in a format that is compatible with CSS and with Pango attributes.

Flags: Read

Default value: ""

Since: 3.22.30

The “language” property

“language” gchar *

The language for which the “font-features” were selected, in a format that is compatible with CSS and with Pango attributes.

Flags: Read / Write

Default value: ""

Since: 3.22.30

The “level” property

“level” GtkFontChooserLevel

The level of granularity to offer for selecting fonts.

Flags: Read / Write

Default value: GTK_FONT_CHOOSER_LEVEL_STYLE | GTK_FONT_CHOOSER_LEVEL_SIZE

Since: 3.22.30

The “preview-text” property

```
  “preview-text”              gchar *
```

The string with which to preview the font.

Flags: Read / Write

Default value: "The quick brown fox jumps over the lazy dog."

The “show-preview-entry” property

```
  “show-preview-entry”        gboolean
```

Whether to show an entry to change the preview text.

Flags: Read / Write

Default value: TRUE

Signal Details

The “font-activated” signal

```
void
user_function (GtkFontChooser *self,
               gchar          *fontname,
               gpointer        user_data)
```

Emitted when a font is activated. This usually happens when the user double clicks an item, or an item is selected and the user presses one of the keys Space, Shift+Space, Return or Enter.

Parameters

self	the object which received the signal
fontname	the font name
user_data	user data set when the signal handler was connected.

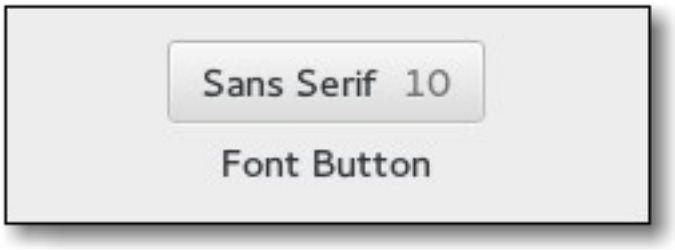
Flags: Run First

See Also

GtkFontChooserDialog, GtkFontChooserWidget, GtkFontButton

GtkFontButton

GtkFontButton — A button to launch a font chooser dialog



Functions

GtkWidget *	gtk_font_button_new ()
GtkWidget *	gtk_font_button_new_with_font ()
gboolean	gtk_font_button_set_font_name ()
const gchar *	gtk_font_button_get_font_name ()
void	gtk_font_button_set_show_style ()
gboolean	gtk_font_button_get_show_style ()
void	gtk_font_button_set_show_size ()
gboolean	gtk_font_button_get_show_size ()
void	gtk_font_button_set_use_font ()
gboolean	gtk_font_button_get_use_font ()
void	gtk_font_button_set_use_size ()
gboolean	gtk_font_button_get_use_size ()
void	gtk_font_button_set_title ()
const gchar *	gtk_font_button_get_title ()

Properties

gchar *	font-name	Read / Write
gboolean	show-size	Read / Write
gboolean	show-style	Read / Write
gchar *	title	Read / Write
gboolean	use-font	Read / Write
gboolean	use-size	Read / Write

Signals

void	font-set	Run First

Types and Values

struct	GtkFontButton

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkContainer
            └── GtkBin
                └── GtkButton
                    └── GtkFontButton
```

Implemented Interfaces

GtkFontButton implements AtkImplementorIface, GtkBuildable, GtkActionable, GtkActivatable and GtkFontChooser.

Includes

```
#include <gtk/gtk.h>
```

Description

The GtkFontButton is a button which displays the currently selected font an allows to open a font chooser dialog to change the font. It is suitable widget for selecting a font in a preference dialog.

CSS nodes

GtkFontButton has a single CSS node with name button and style class .font.

Functions

gtk_font_button_new ()

```
GtkWidget *
gtk_font_button_new (void);
```

Creates a new font picker widget.

Returns

a new font picker widget.

Since: 2.4

---

gtk_font_button_new_with_font ()

```
GtkWidget *
gtk_font_button_new_with_font (const gchar *fontname);
```

Creates a new font picker widget.

Parameters

fontname	Name of font to display in font chooser dialog

Returns

a new font picker widget.

Since: 2.4

---

gtk_font_button_set_font_name ()

```
gboolean
gtk_font_button_set_font_name (GtkFontButton *font_button,
                               const gchar *fontname);
```

`gtk_font_button_set_font_name` has been deprecated since version 3.22 and should not be used in newly-written code.

Use `gtk_font_chooser_set_font()` instead

Sets or updates the currently-displayed font in font picker dialog.

Parameters

font_button	a GtkFontButton
fontname	Name of font to display in font chooser dialog

Returns

`TRUE`

Since: 2.4

---

gtk_font_button_get_font_name ()

```
const gchar *
gtk_font_button_get_font_name (GtkFontButton *font_button);
```

`gtk_font_button_get_font_name` has been deprecated since version 3.22 and should not be used in newly-written code.

Use `gtk_font_chooser_get_font()` instead

Retrieves the name of the currently selected font. This name includes style and size information as well. If you want to render something with the font, use this string with `pango_font_description_from_string()` . If you're interested in peeking certain values (family name, style, size, weight) just query these properties from the PangoFontDescription object.

Parameters

font_button	a GtkFontButton

Returns

an internal copy of the font name which must not be freed.

Since: 2.4

---

gtk_font_button_set_show_style ()

```
void
gtk_font_button_set_show_style (GtkFontButton *font_button,
                                gboolean show_style);
```

If `show_style` is `TRUE`, the font style will be displayed along with name of the selected font.

Parameters

font_button	a GtkFontButton
show_style	`TRUE` if font style should be displayed in label.

Since: 2.4

---

gtk_font_button_get_show_style ()

```
gboolean
gtk_font_button_get_show_style (GtkFontButton *font_button);
```

Returns whether the name of the font style will be shown in the label.

Parameters

font_button	a GtkFontButton

Returns

whether the font style will be shown in the label.

Since: 2.4

---

gtk_font_button_set_show_size ()

```
void
gtk_font_button_set_show_size (GtkFontButton *font_button,
                               gboolean show_size);
```

If `show_size` is `TRUE`, the font size will be displayed along with the name of the selected font.

Parameters

font_button	a GtkFontButton
show_size	TRUE if font size should be displayed in dialog.

Since: 2.4

---

gtk_font_button_get_show_size ()

```
gboolean
gtk_font_button_get_show_size (GtkFontButton *font_button);
```

Returns whether the font size will be shown in the label.

Parameters

font_button	a GtkFontButton

Returns

whether the font size will be shown in the label.

Since: 2.4

---

gtk_font_button_set_use_font ()

```
void
gtk_font_button_set_use_font (GtkFontButton *font_button,
                              gboolean use_font);
```

If use_font is TRUE, the font name will be written using the selected font.

Parameters

font_button	a GtkFontButton
use_font	If TRUE, font name will be written using font chosen.

Since: 2.4

---

gtk_font_button_get_use_font ()

```
gboolean
gtk_font_button_get_use_font (GtkFontButton *font_button);
```

Returns whether the selected font is used in the label.

Parameters

font_button	a GtkFontButton

Returns

whether the selected font is used in the label.

Since: 2.4

---

gtk_font_button_set_use_size ()

```
void
gtk_font_button_set_use_size (GtkFontButton *font_button,
                              gboolean use_size);
```

If `use_size` is `TRUE`, the font name will be written using the selected size.

Parameters

font_button	a GtkFontButton
use_size	If `TRUE`, font name will be written using the selected size.

Since: 2.4

---

gtk_font_button_get_use_size ()

```
gboolean
gtk_font_button_get_use_size (GtkFontButton *font_button);
```

Returns whether the selected size is used in the label.

Parameters

font_button	a GtkFontButton

Returns

whether the selected size is used in the label.

Since: 2.4

---

gtk_font_button_set_title ()

```
void
gtk_font_button_set_title (GtkFontButton *font_button,
                           const gchar *title);
```

Sets the title for the font chooser dialog.

Parameters

font_button	a GtkFontButton
title	a string containing the font chooser dialog title

Since: 2.4

---

gtk_font_button_get_title ()

```
const gchar *
gtk_font_button_get_title (GtkFontButton *font_button);
```

Retrieves the title of the font chooser dialog.

Parameters

font_button	a GtkFontButton

Returns

an internal copy of the title string which must not be freed.

Since: 2.4

Types and Values

struct GtkFontButton

```
struct GtkFontButton;
```

Property Details

The “font-name” property

```
  “font-name”                gchar *
```

The name of the currently selected font.

`GtkFontButton:font-name` has been deprecated since version 3.22 and should not be used in newly-written code.

Use the “font” property instead

Flags: Read / Write

Default value: "Sans 12"

Since: 2.4

The “show-size” property

“show-size” gboolean

If this property is set to TRUE, the selected font size will be shown in the label. For a more WYSIWYG way to show the selected size, see the ::use-size property.

Flags: Read / Write

Default value: TRUE

Since: 2.4

The “show-style” property

“show-style” gboolean

If this property is set to TRUE, the name of the selected font style will be shown in the label. For a more WYSIWYG way to show the selected style, see the ::use-font property.

Flags: Read / Write

Default value: TRUE

Since: 2.4

The “title” property

“title” gchar *

The title of the font chooser dialog.

Flags: Read / Write

Default value: "Pick a Font"

Since: 2.4

The “use-font” property

“use-font” gboolean

If this property is set to TRUE, the label will be drawn in the selected font.

Flags: Read / Write

Default value: FALSE

Since: 2.4

The “use-size” property

```
  “use-size”                 gboolean
```

If this property is set to TRUE, the label will be drawn with the selected font size.

Flags: Read / Write

Default value: FALSE

Since: 2.4

Signal Details

The “font-set” signal

```
void
user_function (GtkFontButton *widget,
               gpointer       user_data)
```

The ::font-set signal is emitted when the user selects a font. When handling this signal, use gtk_font_chooser_get_font() to find out which font was just selected.

Note that this signal is only emitted when the user changes the font. If you need to react to programmatic font changes as well, use the notify::font signal.

Parameters

widget	the object which received the signal.
user_data	user data set when the signal handler was connected.

Flags: Run First

Since: 2.4

See Also

GtkFontChooserDialog, GtkColorButton.

GtkFontChooserWidget

GtkFontChooserWidget — A widget for selecting fonts

Functions

GtkWidget *	gtk_font_chooser_widget_new ()

Properties

GAction *	tweak-action	Read

Types and Values

struct	GtkFontChooserWidget
struct	GtkFontChooserWidgetClass

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkContainer
            └── GtkBox
                └── GtkFontChooserWidget
```

Implemented Interfaces

GtkFontChooserWidget implements AtkImplementorIface, GtkBuildable, GtkOrientable and GtkFontChooser.

Includes

```
#include <gtk/gtk.h>
```

Description

The GtkFontChooserWidget widget lists the available fonts, styles and sizes, allowing the user to select a font. It is used in the GtkFontChooserDialog widget to provide a dialog box for selecting fonts.

To set the font which is initially selected, use `gtk_font_chooser_set_font()` or `gtk_font_chooser_set_font_desc()`.

To get the selected font use `gtk_font_chooser_get_font()` or `gtk_font_chooser_get_font_desc()`.

To change the text which is shown in the preview area, use `gtk_font_chooser_set_preview_text()`.

CSS nodes

GtkFontChooserWidget has a single CSS node with name fontchooser.

Functions

gtk_font_chooser_widget_new ()

```
GtkWidget *
gtk_font_chooser_widget_new (void);
```

Creates a new GtkFontChooserWidget.

Returns

a new GtkFontChooserWidget

Since: 3.2

Types and Values

struct GtkFontChooserWidget

```
struct GtkFontChooserWidget;
```

---

struct GtkFontChooserWidgetClass

```
struct GtkFontChooserWidgetClass {
  GtkBoxClass parent_class;
};
```

Members

Property Details

The “tweak-action” property

```
  “tweak-action”             GAction *
```

A toggle action that can be used to switch to the tweak page of the font chooser widget, which lets the user tweak the OpenType features and variation axes of the selected font.

The action will be enabled or disabled depending on whether the selected font has any features or axes.

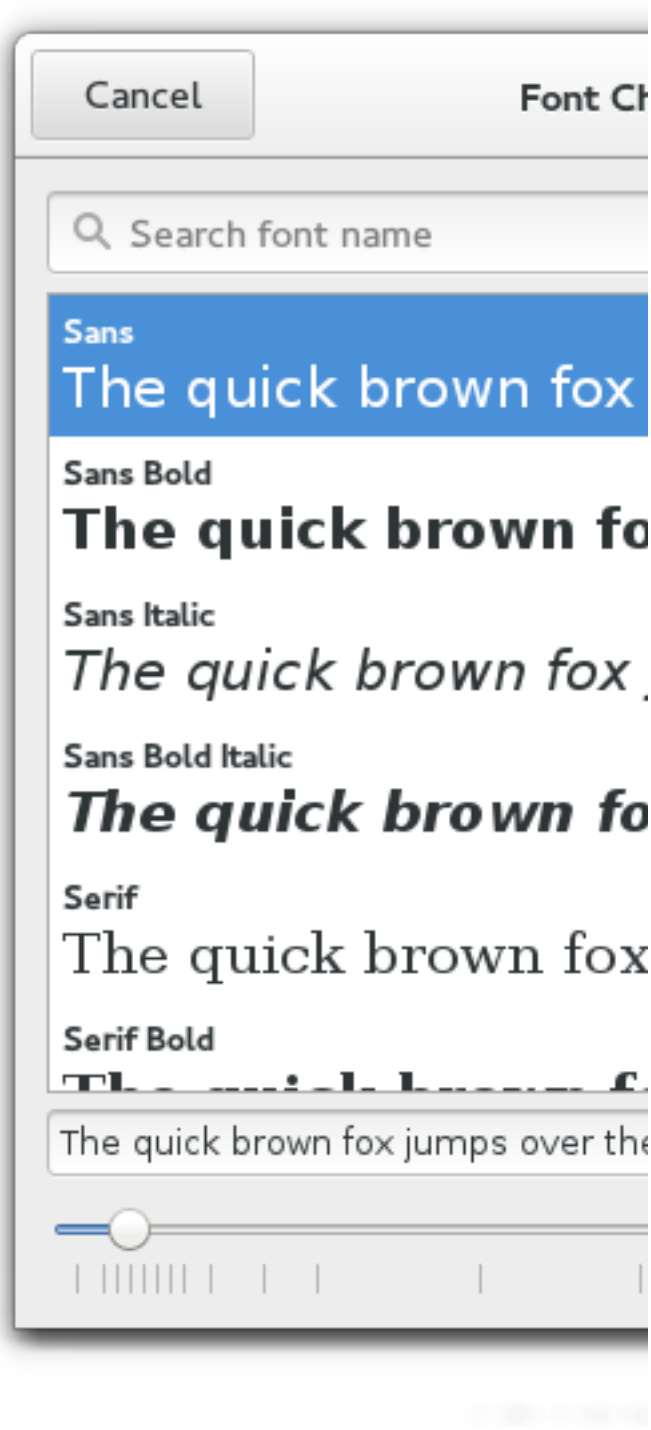
Flags: Read

See Also

GtkFontChooserDialog

GtkFontChooserDialog

GtkFontChooserDialog — A dialog for selecting fonts



Functions

GtkWidget *	gtk_font_chooser_dialog_new ()

Types and Values

struct	GtkFontChooserDialog
struct	GtkFontChooserDialogClass

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkContainer
            └── GtkBin
                └── GtkWindow
                    └── GtkDialog
                        └── GtkFontChooserDialog
```

Implemented Interfaces

GtkFontChooserDialog implements AtkImplementorIface, GtkBuildable and GtkFontChooser.

Includes

```
#include <gtk/gtk.h>
```

Description

The GtkFontChooserDialog widget is a dialog for selecting a font. It implements the GtkFontChooser interface.

GtkFontChooserDialog as GtkBuildable

The GtkFontChooserDialog implementation of the GtkBuildable interface exposes the buttons with the names “select_button” and “cancel_button”.

Functions

gtk_font_chooser_dialog_new ()

```
GtkWidget *
gtk_font_chooser_dialog_new (const gchar *title,
                             GtkWindow *parent);
```

Creates a new GtkFontChooserDialog.

Parameters

title	Title of the dialog, or `NULL`.	[allow-none]
parent	Transient parent of the dialog, or `NULL`.	[allow-none]

Returns

a new GtkFontChooserDialog

Since: 3.2

Types and Values

struct GtkFontChooserDialog

```
struct GtkFontChooserDialog;
```

---

struct GtkFontChooserDialogClass

```
struct GtkFontChooserDialogClass {
  GtkDialogClass parent_class;
};
```

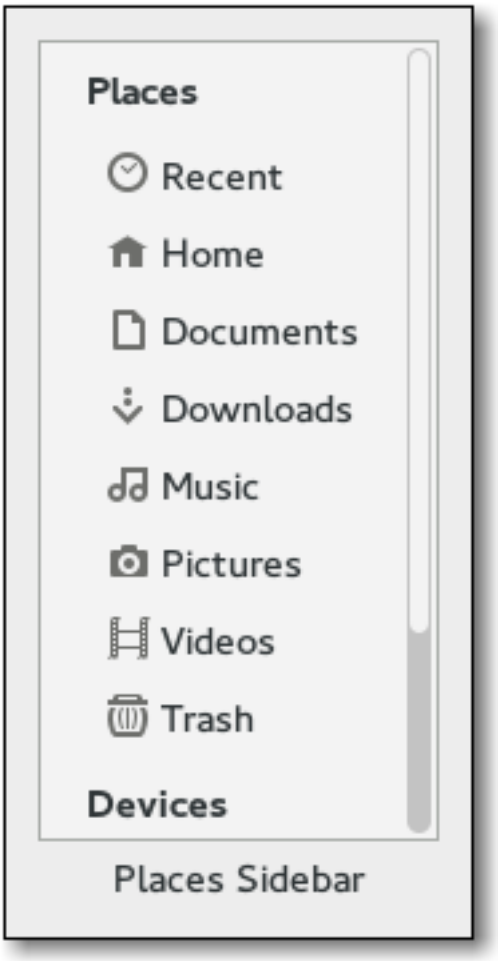
Members

See Also

GtkFontChooser, GtkDialog

GtkPlacesSidebar

GtkPlacesSidebar — Sidebar that displays frequently-used places in the file system



Functions

GtkWidget *	gtk_places_sidebar_new ()
void	gtk_places_sidebar_set_open_flags ()
GtkPlacesOpenFlags	gtk_places_sidebar_get_open_flags ()
void	gtk_places_sidebar_set_location ()
GFile *	gtk_places_sidebar_get_location ()
void	gtk_places_sidebar_set_show_recent ()
gboolean	gtk_places_sidebar_get_show_recent ()
void	gtk_places_sidebar_set_show_desktop ()
gboolean	gtk_places_sidebar_get_show_desktop ()
void	gtk_places_sidebar_add_shortcut ()
void	gtk_places_sidebar_remove_shortcut ()
GSList *	gtk_places_sidebar_list_shortcuts ()
GFile *	gtk_places_sidebar_get_nth_bookmark ()
gboolean	gtk_places_sidebar_get_show_connect_to_server ()
void	gtk_places_sidebar_set_show_connect_to_server ()
gboolean	gtk_places_sidebar_get_local_only ()
void	gtk_places_sidebar_set_local_only ()
gboolean	gtk_places_sidebar_get_show_enter_location ()
void	gtk_places_sidebar_set_show_enter_location ()
gboolean	gtk_places_sidebar_get_show_trash ()
void	gtk_places_sidebar_set_show_trash ()
gboolean	gtk_places_sidebar_get_show_other_locations ()

void	gtk_places_sidebar_set_show_other_locations ()
void	gtk_places_sidebar_set_drop_targets_visible ()

Properties

gboolean	local-only	Read / Write
GFile *	location	Read / Write
GtkPlacesOpenFlags	open-flags	Read / Write
gboolean	populate-all	Read / Write
gboolean	show-connect-to-server	Read / Write
gboolean	show-desktop	Read / Write
gboolean	show-enter-location	Read / Write
gboolean	show-other-locations	Read / Write
gboolean	show-recent	Read / Write
gboolean	show-starred-location	Read / Write
gboolean	show-trash	Read / Write

Signals

gint	drag-action-ask	Run Last
gint	drag-action-requested	Run Last
void	drag-perform-drop	Run First
void	mount	Run First
void	open-location	Run First
void	populate-popup	Run First
void	show-connect-to-server	Run First
void	show-enter-location	Run First
void	show-error-message	Run First
void	show-other-locations	Run First
void	show-other-locations-with-flags	Run First
void	show-starred-location	Run First
void	unmount	Run First

Types and Values

	GtkPlacesSidebar
enum	GtkPlacesOpenFlags

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkContainer
            └── GtkBin
                └── GtkScrolledWindow
                    └── GtkPlacesSidebar
```

Implemented Interfaces

GtkPlacesSidebar implements AtkImplementorIface and GtkBuildable.

Includes

```
#include <gtk/gtk.h>
```

Description

GtkPlacesSidebar is a widget that displays a list of frequently-used places in the file system: the user’s home directory, the user’s bookmarks, and volumes and drives. This widget is used as a sidebar in GtkFileChooser and may be used by file managers and similar programs.

The places sidebar displays drives and volumes, and will automatically mount or unmount them when the user selects them.

Applications can hook to various signals in the places sidebar to customize its behavior. For example, they can add extra commands to the context menu of the sidebar.

While bookmarks are completely in control of the user, the places sidebar also allows individual applications to provide extra shortcut folders that are unique to each application. For example, a Paint program may want to add a shortcut for a Clipart folder. You can do this with `gtk_places_sidebar_add_shortcut()`.

To make use of the places sidebar, an application at least needs to connect to the “open-location” signal. This is emitted when the user selects in the sidebar a location to open. The application should also call `gtk_places_sidebar_set_location()` when it changes the currently-viewed location.

CSS nodes

GtkPlacesSidebar uses a single CSS node with name placessidebar and style class .sidebar.

Among the children of the places sidebar, the following style classes can be used:

- .sidebar-new-bookmark-row for the 'Add new bookmark' row
- .sidebar-placeholder-row for a row that is a placeholder
- .has-open-popup when a popup is open for a row

Functions

gtk_places_sidebar_new ()

```
GtkWidget *
gtk_places_sidebar_new (void);
```

Creates a new GtkPlacesSidebar widget.

The application should connect to at least the “open-location” signal to be notified when the user makes a selection in the sidebar.

Returns

a newly created GtkPlacesSidebar

Since: 3.10

---

gtk_places_sidebar_set_open_flags ()

```
void
gtk_places_sidebar_set_open_flags (GtkPlacesSidebar *sidebar,
                                   GtkPlacesOpenFlags flags);
```

Sets the way in which the calling application can open new locations from the places sidebar. For example, some applications only open locations “directly” into their main view, while others may support opening locations in a new notebook tab or a new window.

This function is used to tell the places `sidebar` about the ways in which the application can open new locations, so that the sidebar can display (or not) the “Open in new tab” and “Open in new window” menu items as appropriate.

When the “open-location” signal is emitted, its flags argument will be set to one of the `flags` that was passed in `gtk_places_sidebar_set_open_flags()`.

Passing 0 for `flags` will cause GTK_PLACES_OPEN_NORMAL to always be sent to callbacks for the “open-location” signal.

Parameters

sidebar	a places sidebar
flags	Bitmask of modes in which the calling application can open locations

Since: 3.10

---

gtk_places_sidebar_get_open_flags ()

```
GtkPlacesOpenFlags
gtk_places_sidebar_get_open_flags (GtkPlacesSidebar *sidebar);
```

Gets the open flags.

Parameters

sidebar	a GtkPlacesSidebar

Returns

the GtkPlacesOpenFlags of `sidebar`

Since: 3.10

---

gtk_places_sidebar_set_location ()

```
void
gtk_places_sidebar_set_location (GtkPlacesSidebar *sidebar,
                                 GFile *location);
```

Sets the location that is being shown in the widgets surrounding the `sidebar` , for example, in a folder view in a file manager. In turn, the `sidebar` will highlight that location if it is being shown in the list of places, or it will unhighlight everything if the `location` is not among the places in the list.

Parameters

sidebar	a places sidebar	
location	location to select, or `NULL` for no current path.	[nullable]

Since: 3.10

---

gtk_places_sidebar_get_location ()

```
GFile *
gtk_places_sidebar_get_location (GtkPlacesSidebar *sidebar);
```

Gets the currently selected location in the `sidebar` . This can be `NULL` when nothing is selected, for example, when `gtk_places_sidebar_set_location()` has been called with a location that is not among the sidebar's list of places to show.

You can use this function to get the selection in the `sidebar` . Also, if you connect to the "populate-popup" signal, you can use this function to get the location that is being referred to during the callbacks for your menu items.

Parameters

sidebar	a places sidebar

Returns

a GFile with the selected location, or `NULL` if nothing is visually selected.

[nullable][transfer full]

Since: 3.10

---

gtk_places_sidebar_set_show_recent ()

```
void
gtk_places_sidebar_set_show_recent (GtkPlacesSidebar *sidebar,
                                    gboolean show_recent);
```

Sets whether the `sidebar` should show an item for recent files. The default value for this option is determined

by the desktop environment, but this function can be used to override it on a per-application basis.

Parameters

sidebar	a places sidebar
show_recent	whether to show an item for recent files

Since: 3.18

---

gtk_places_sidebar_get_show_recent ()

```
gboolean
gtk_places_sidebar_get_show_recent (GtkPlacesSidebar *sidebar);
```

Returns the value previously set with `gtk_places_sidebar_set_show_recent()`

Parameters

sidebar	a places sidebar

Returns

`TRUE` if the sidebar will display a builtin shortcut for recent files

Since: 3.18

---

gtk_places_sidebar_set_show_desktop ()

```
void
gtk_places_sidebar_set_show_desktop (GtkPlacesSidebar *sidebar,
                                     gboolean show_desktop);
```

Sets whether the `sidebar` should show an item for the Desktop folder. The default value for this option is determined by the desktop environment and the user’s configuration, but this function can be used to override it on a per-application basis.

Parameters

sidebar	a places sidebar
show_desktop	whether to show an item for the Desktop folder

Since: 3.10

---

gtk_places_sidebar_get_show_desktop ()

```
gboolean
```

```
gtk_places_sidebar_get_show_desktop (GtkPlacesSidebar *sidebar);
```

Returns the value previously set with `gtk_places_sidebar_set_show_desktop()`

Parameters

sidebar	a places sidebar

Returns

`TRUE` if the sidebar will display a builtin shortcut to the desktop folder.

Since: 3.10

---

gtk_places_sidebar_add_shortcut ()

```
void
gtk_places_sidebar_add_shortcut (GtkPlacesSidebar *sidebar,
                                 GFile *location);
```

Applications may want to present some folders in the places sidebar if they could be immediately useful to users. For example, a drawing program could add a “/usr/share/clipart” location when the sidebar is being used in an “Insert Clipart” dialog box.

This function adds the specified `location` to a special place for immutable shortcuts. The shortcuts are application-specific; they are not shared across applications, and they are not persistent. If this function is called multiple times with different locations, then they are added to the sidebar’s list in the same order as the function is called.

Parameters

sidebar	a places sidebar
location	location to add as an application-specific shortcut

Since: 3.10

---

gtk_places_sidebar_remove_shortcut ()

```
void
gtk_places_sidebar_remove_shortcut (GtkPlacesSidebar *sidebar,
                                    GFile *location);
```

Removes an application-specific shortcut that has been previously been inserted with `gtk_places_sidebar_add_shortcut()`. If the `location` is not a shortcut in the sidebar, then nothing is done.

Parameters

sidebar	a places sidebar
location	location to remove

Since: 3.10

---

gtk_places_sidebar_list_shortcuts ()

```
GSList *
gtk_places_sidebar_list_shortcuts (GtkPlacesSidebar *sidebar);
```

Gets the list of shortcuts.

Parameters

sidebar	a places sidebar

Returns

A GSList of GFile of the locations that have been added as application-specific shortcuts with `gtk_places_sidebar_add_shortcut()`. To free this list, you can use

```
1                                       g_slist_free_full (list, (GDestroyNotify)
                                        g_object_unref);
```

.

[element-type GFile][transfer full]

Since: 3.10

---

gtk_places_sidebar_get_nth_bookmark ()

```
GFile *
gtk_places_sidebar_get_nth_bookmark (GtkPlacesSidebar *sidebar,
                                     gint n);
```

This function queries the bookmarks added by the user to the places sidebar, and returns one of them. This function is used by GtkFileChooser to implement the “Alt-1”, “Alt-2”, etc. shortcuts, which activate the cooresponding bookmark.

Parameters

sidebar	a places sidebar
n	index of the bookmark to query

Returns

The bookmark specified by the index `n` , or `NULL` if no such index exist. Note that the indices start at 0, even though the file chooser starts them with the keyboard shortcut "Alt-1".

[nullable][transfer full]

Since: 3.10

---

gtk_places_sidebar_get_show_connect_to_server ()

```
gboolean
gtk_places_sidebar_get_show_connect_to_server
                                (GtkPlacesSidebar *sidebar);
```

`gtk_places_sidebar_get_show_connect_to_server` has been deprecated since version 3.18 and should not be used in newly-written code.

It is recommended to group this functionality with the drives and network location under the new 'Other Location' item

Returns the value previously set with `gtk_places_sidebar_set_show_connect_to_server()`

Parameters

sidebar	a places sidebar

Returns

`TRUE` if the sidebar will display a “Connect to Server” item.

---

gtk_places_sidebar_set_show_connect_to_server ()

```
void
gtk_places_sidebar_set_show_connect_to_server
                                (GtkPlacesSidebar *sidebar,
                                 gboolean show_connect_to_server);
```

`gtk_places_sidebar_set_show_connect_to_server` has been deprecated since version 3.18 and should not be used in newly-written code.

It is recommended to group this functionality with the drives and network location under the new 'Other Location' item

Sets whether the `sidebar` should show an item for connecting to a network server; this is off by default. An application may want to turn this on if it implements a way for the user to connect to network servers directly.

If you enable this, you should connect to the “show-connect-to-server” signal.

Parameters

sidebar	a places sidebar
show_connect_to_server	whether to show an item for the Connect to Server command

Since: 3.10

---

gtk_places_sidebar_get_local_only ()

```
gboolean
gtk_places_sidebar_get_local_only (GtkPlacesSidebar *sidebar);
```

Returns the value previously set with gtk_places_sidebar_set_local_only().

Parameters

sidebar	a places sidebar

Returns

TRUE if the sidebar will only show local files.

Since: 3.12

---

gtk_places_sidebar_set_local_only ()

```
void
gtk_places_sidebar_set_local_only (GtkPlacesSidebar *sidebar,
                                   gboolean local_only);
```

Sets whether the sidebar should only show local files.

Parameters

sidebar	a places sidebar
local_only	whether to show only local files

Since: 3.12

---

gtk_places_sidebar_get_show_enter_location ()

```
gboolean
gtk_places_sidebar_get_show_enter_location
                                (GtkPlacesSidebar *sidebar);
```

Returns the value previously set with gtk_places_sidebar_set_show_enter_location()

Parameters

sidebar	a places sidebar

Returns

TRUE if the sidebar will display an “Enter Location” item.

Since: 3.14

---

gtk_places_sidebar_set_show_enter_location ()

```
void
gtk_places_sidebar_set_show_enter_location
                                (GtkPlacesSidebar *sidebar,
                                 gboolean show_enter_location);
```

Sets whether the `sidebar` should show an item for entering a location; this is off by default. An application may want to turn this on if manually entering URLs is an expected user action.

If you enable this, you should connect to the "show-enter-location" signal.

Parameters

sidebar	a places sidebar
show_enter_location	whether to show an item to enter a location

Since: 3.14

---

gtk_places_sidebar_get_show_trash ()

```
gboolean
gtk_places_sidebar_get_show_trash (GtkPlacesSidebar *sidebar);
```

Returns the value previously set with `gtk_places_sidebar_set_show_trash()`

Parameters

sidebar	a places sidebar

Returns

`TRUE` if the sidebar will display a "Trash" item.

Since: 3.18

---

gtk_places_sidebar_set_show_trash ()

```
void
gtk_places_sidebar_set_show_trash (GtkPlacesSidebar *sidebar,
                                   gboolean show_trash);
```

Sets whether the `sidebar` should show an item for the Trash location.

Parameters

sidebar	a places sidebar
show_trash	whether to show an item for the

	Trash location

Since: 3.18

---

gtk_places_sidebar_get_show_other_locations ()

```
gboolean
gtk_places_sidebar_get_show_other_locations
                                (GtkPlacesSidebar *sidebar);
```

Returns the value previously set with `gtk_places_sidebar_set_show_other_locations()`

Parameters

sidebar	a places sidebar

Returns

`TRUE` if the sidebar will display an “Other Locations” item.

Since: 3.18

---

gtk_places_sidebar_set_show_other_locations ()

```
void
gtk_places_sidebar_set_show_other_locations
                                (GtkPlacesSidebar *sidebar,
                                 gboolean show_other_locations);
```

Sets whether the `sidebar` should show an item for the application to show an Other Locations view; this is off by default. When set to `TRUE`, persistent devices such as hard drives are hidden, otherwise they are shown in the sidebar. An application may want to turn this on if it implements a way for the user to see and interact with drives and network servers directly.

If you enable this, you should connect to the “show-other-locations” signal.

Parameters

sidebar	a places sidebar
show_other_locations	whether to show an item for the Other Locations view

Since: 3.18

---

gtk_places_sidebar_set_drop_targets_visible ()

```
void
gtk_places_sidebar_set_drop_targets_visible
                                (GtkPlacesSidebar *sidebar,
                                 gboolean visible,
```

```
                                GdkDragContext *context);
```

Make the GtkPlacesSidebar show drop targets, so it can show the available drop targets and a "new bookmark" row. This improves the Drag-and-Drop experience of the user and allows applications to show all available drop targets at once.

This needs to be called when the application is aware of an ongoing drag that might target the sidebar. The drop-targets-visible state will be unset automatically if the drag finishes in the GtkPlacesSidebar. You only need to unset the state when the drag ends on some other widget on your application.

Parameters

sidebar	a places sidebar.
visible	whether to show the valid targets or not.
context	drag context used to ask the source about the action that wants to perform, so hints are more accurate.

Since: 3.18

Types and Values

GtkPlacesSidebar

```
typedef struct _GtkPlacesSidebar GtkPlacesSidebar;
```

---

enum GtkPlacesOpenFlags

These flags serve two purposes. First, the application can call `gtk_places_sidebar_set_open_flags()` using these flags as a bitmask. This tells the sidebar that the application is able to open folders selected from the sidebar in various ways, for example, in new tabs or in new windows in addition to the normal mode.

Second, when one of these values gets passed back to the application in the "open-location" signal, it means that the application should open the selected location in the normal way, in a new tab, or in a new window. The sidebar takes care of determining the desired way to open the location, based on the modifier keys that the user is pressing at the time the selection is made.

If the application never calls `gtk_places_sidebar_set_open_flags()`, then the sidebar will only use GTK_PLACES_OPEN_NORMAL in the "open-location" signal. This is the default mode of operation.

Members

GTK_PLACES_OPEN_NORMAL	This is the default mode that GtkPlacesSidebar uses if no other flags are specified. It indicates that the calling application should open the selected location in the normal way, for example, in the folder view

	beside the sidebar.
GTK_PLACES_OPEN_NEW_TAB	When passed to gtk_places_sidebar_set_open_flags(), this indicates that the application can open folders selected from the sidebar in new tabs. This value will be passed to the "open-location" signal when the user selects that a location be opened in a new tab instead of in the standard fashion.
GTK_PLACES_OPEN_NEW_WINDOW	Similar to GTK_PLACES_OPEN_NEW_TAB , but indicates that the application can open folders in new windows.

Property Details

The "local-only" property

"local-only" gboolean

Whether the sidebar only includes local files.

Flags: Read / Write

Default value: FALSE

The "location" property

"location" GFile *

The location to highlight in the sidebar.

Flags: Read / Write

The "open-flags" property

"open-flags" GtkPlacesOpenFlags

Modes in which the calling application can open locations selected in the sidebar.

Flags: Read / Write

Default value: GTK_PLACES_OPEN_NORMAL

The "populate-all" property

"populate-all" gboolean

If :populate-all is TRUE, the “populate-popup” signal is also emitted for popovers.

Flags: Read / Write

Default value: FALSE

Since: 3.18

The “show-connect-to-server” property

“show-connect-to-server” gboolean

Whether the sidebar includes a builtin shortcut to a 'Connect to server' dialog.

Flags: Read / Write

Default value: FALSE

The “show-desktop” property

“show-desktop” gboolean

Whether the sidebar includes a builtin shortcut to the Desktop folder.

Flags: Read / Write

Default value: TRUE

The “show-enter-location” property

“show-enter-location” gboolean

Whether the sidebar includes a builtin shortcut to manually enter a location.

Flags: Read / Write

Default value: FALSE

The “show-other-locations” property

“show-other-locations” gboolean

Whether the sidebar includes an item to show external locations.

Flags: Read / Write

Default value: FALSE

The “show-recent” property

“show-recent” gboolean

Whether the sidebar includes a builtin shortcut for recent files.

Flags: Read / Write

Default value: TRUE

The “show-starred-location” property

“show-starred-location” gboolean

Whether the sidebar includes an item to show starred files.

Flags: Read / Write

Default value: FALSE

The “show-trash” property

“show-trash” gboolean

Whether the sidebar includes a builtin shortcut to the Trash location.

Flags: Read / Write

Default value: TRUE

Signal Details

The “drag-action-ask” signal

```
gint
user_function (GtkPlacesSidebar *sidebar,
               gint              actions,
               gpointer          user_data)
```

The places sidebar emits this signal when it needs to ask the application to pop up a menu to ask the user for which drag action to perform.

Parameters

sidebar	the object which received the signal.
actions	Possible drag actions that need to be asked for.
user_data	user data set when the signal handler was connected.

Returns

the final drag action that the sidebar should pass to the drag side of the drag-and-drop operation.

Flags: Run Last

Since: 3.10

The `"drag-action-requested"` signal

```
gint
user_function (GtkPlacesSidebar *sidebar,
               GdkDragContext   *context,
               GObject          *dest_file,
               gpointer          source_file_list,
               gpointer          user_data)
```

When the user starts a drag-and-drop operation and the sidebar needs to ask the application for which drag action to perform, then the sidebar will emit this signal.

The application can evaluate the `context` for customary actions, or it can check the type of the files indicated by `source_file_list` against the possible actions for the destination `dest_file` .

The drag action to use must be the return value of the signal handler.

Parameters

sidebar	the object which received the signal.	
context	GdkDragContext with information about the drag operation.	[type Gdk.DragContext]
dest_file	GFile with the tentative location that is being hovered for a drop.	[type Gio.File]
source_file_list	List of GFile that are being dragged.	[type GLib.List][element-type GFile][transfer none]
user_data	user data set when the signal handler was connected.	

Returns

The drag action to use, for example, GDK_ACTION_COPY or GDK_ACTION_MOVE, or 0 if no action is allowed here (i.e. drops are not allowed in the specified `dest_file`).

Flags: Run Last

Since: 3.10

The `"drag-perform-drop"` signal

```
void
user_function (GtkPlacesSidebar *sidebar,
               GObject          *dest_file,
               gpointer          source_file_list,
```

```
                gint              action,
                gpointer          user_data)
```

The places sidebar emits this signal when the user completes a drag-and-drop operation and one of the sidebar's items is the destination. This item is in the `dest_file` , and the `source_file_list` has the list of files that are dropped into it and which should be copied/moved/etc. based on the specified `action` .

Parameters

sidebar	the object which received the signal.	
dest_file	Destination GFile.	[type Gio.File]
source_file_list	GList of GFile that got dropped.	[type GLib.List][element-type GFile][transfer none]
action	Drop action to perform.	
user_data	user data set when the signal handler was connected.	

Flags: Run First

Since: 3.10

The `"mount"` signal

```
void
user_function (GtkPlacesSidebar *sidebar,
               GMountOperation  *mount_operation,
               gpointer          user_data)
```

The places sidebar emits this signal when it starts a new operation because the user clicked on some location that needs mounting. In this way the application using the GtkPlacesSidebar can track the progress of the operation and, for example, show a notification.

Parameters

sidebar	the object which received the signal.
mount_operation	the GMountOperation that is going to start.
user_data	user data set when the signal handler was connected.

Flags: Run First

Since: 3.20

The `"open-location"` signal

```
void
user_function (GtkPlacesSidebar  *sidebar,
               GObject           *location,
               GtkPlacesOpenFlags open_flags,
               gpointer           user_data)
```

The places sidebar emits this signal when the user selects a location in it. The calling application should display

the contents of that location; for example, a file manager should show a list of files in the specified location.

Parameters

sidebar	the object which received the signal.	
location	GFile to which the caller should switch.	[type Gio.File]
open_flags	a single value from GtkPlacesOpenFlags specifying how the `location` should be opened.	
user_data	user data set when the signal handler was connected.	

Flags: Run First

Since: 3.10

The “populate-popup” signal

```
void
user_function (GtkPlacesSidebar *sidebar,
               GtkWidget        *container,
               GFile            *selected_item,
               GVolume          *selected_volume,
               gpointer          user_data)
```

The places sidebar emits this signal when the user invokes a contextual popup on one of its items. In the signal handler, the application may add extra items to the menu as appropriate. For example, a file manager may want to add a "Properties" command to the menu.

It is not necessary to store the `selected_item` for each menu item; during their callbacks, the application can use `gtk_places_sidebar_get_location()` to get the file to which the item refers.

The `selected_item` argument may be `NULL` in case the selection refers to a volume. In this case, `selected_volume` will be non-`NULL`. In this case, the calling application will have to `g_object_ref()` the `selected_volume` and keep it around to use it in the callback.

The `container` and all its contents are destroyed after the user dismisses the popup. The popup is re-created (and thus, this signal is emitted) every time the user activates the contextual menu.

Before 3.18, the `container` always was a GtkMenu, and you were expected to add your items as GtkMenuItems. Since 3.18, the popup may be implemented as a GtkPopover, in which case `container` will be something else, e.g. a GtkBox, to which you may add GtkModelButtons or other widgets, such as GtkEntries, GtkSpinButtons, etc. If your application can deal with this situation, you can set “populate-all” to `TRUE` to request that this signal is emitted for populating popovers as well.

Parameters

sidebar	the object which received the signal.	
container	a GtkMenu or another GtkContainer.	[type Gtk.Widget]

selected_item	GFile with the item to which the popup should refer, or `NULL` in the case of a `selected_volume` .	[type Gio.File][nullable]
selected_volume	GVolume if the selected item is a volume, or `NULL` if it is a file.	[type Gio.Volume][nullable]
user_data	user data set when the signal handler was connected.	

Flags: Run First

Since: 3.10

The “show-connect-to-server” signal

```
void
user_function (GtkPlacesSidebar *sidebar,
               gpointer          user_data)
```

The places sidebar emits this signal when it needs the calling application to present an way to connect directly to a network server. For example, the application may bring up a dialog box asking for a URL like "sftp://ftp.example.com". It is up to the application to create the corresponding mount by using, for example, `g_file_mount_enclosing_volume()`.

`GtkPlacesSidebar::show-connect-to-server` has been deprecated since version 3.18 and should not be used in newly-written code.

use the “show-other-locations” signal to connect to network servers.

Parameters

sidebar	the object which received the signal.
user_data	user data set when the signal handler was connected.

Flags: Run First

The “show-enter-location” signal

```
void
user_function (GtkPlacesSidebar *sidebar,
               gpointer          user_data)
```

The places sidebar emits this signal when it needs the calling application to present an way to directly enter a location. For example, the application may bring up a dialog box asking for a URL like "http://http.example.com".

Parameters

sidebar	the object which received the signal.
user_data	user data set when the signal handler was connected.

Flags: Run First

Since: 3.14

The “show-error-message” signal

```
void
user_function (GtkPlacesSidebar *sidebar,
               gchar            *primary,
               gchar            *secondary,
               gpointer          user_data)
```

The places sidebar emits this signal when it needs the calling application to present an error message. Most of these messages refer to mounting or unmounting media, for example, when a drive cannot be started for some reason.

Parameters

sidebar	the object which received the signal.
primary	primary message with a summary of the error to show.
secondary	secondary message with details of the error to show.
user_data	user data set when the signal handler was connected.

Flags: Run First

Since: 3.10

The “show-other-locations” signal

```
void
user_function (GtkPlacesSidebar *sidebar,
               gpointer          user_data)
```

The places sidebar emits this signal when it needs the calling application to present a way to show other locations e.g. drives and network access points. For example, the application may bring up a page showing persistent volumes and discovered network addresses.

`GtkPlacesSidebar::show-other-locations` has been deprecated since version 3.20 and should not be used in newly-written code.

use the “show-other-locations-with-flags” which includes the open flags in order to allow the user to specify to open in a new tab or window, in a similar way than “open-location”

Parameters

sidebar	the object which received the signal.
user_data	user data set when the signal handler was connected.

Flags: Run First

Since: 3.18

The “show-other-locations-with-flags” signal

```
void
user_function (GtkPlacesSidebar  *sidebar,
               GtkPlacesOpenFlags open_flags,
               gpointer           user_data)
```

The places sidebar emits this signal when it needs the calling application to present a way to show other locations e.g. drives and network access points. For example, the application may bring up a page showing persistent volumes and discovered network addresses.

Parameters

sidebar	the object which received the signal.
open_flags	a single value from GtkPlacesOpenFlags specifying how it should be opened.
user_data	user data set when the signal handler was connected.

Flags: Run First

Since: 3.20

The “show-starred-location” signal

```
void
user_function (GtkPlacesSidebar  *sidebar,
               GtkPlacesOpenFlags open_flags,
               gpointer           user_data)
```

The places sidebar emits this signal when it needs the calling application to present a way to show the starred files. In GNOME, starred files are implemented by setting the nao:predefined-tag-favorite tag in the tracker database.

Parameters

sidebar	the object which received the signal.
open_flags	a single value from GtkPlacesOpenFlags specifying how the starred file should be opened.
user_data	user data set when the signal handler was connected.

Flags: Run First

Since: 3.22.26

The “unmount” signal

```
void
user_function (GtkPlacesSidebar *sidebar,
               GMountOperation  *mount_operation,
               gpointer          user_data)
```

The places sidebar emits this signal when it starts a new operation because the user for example ejected some drive or unmounted a mount. In this way the application using the GtkPlacesSidebar can track the progress of the operation and, for example, show a notification.

Parameters

sidebar	the object which received the signal.
mount_operation	the GMountOperation that is going to start.
user_data	user data set when the signal handler was connected.

Flags: Run First

Since: 3.20

See Also

GtkFileChooser

Ornaments

GtkFrame — A bin with a decorative frame and optional label

GtkSeparator — A separator widget

GtkFrame

GtkFrame — A bin with a decorative frame and optional label

Functions

GtkWidget *	gtk_frame_new ()

void	gtk_frame_set_label ()
void	gtk_frame_set_label_widget ()
void	gtk_frame_set_label_align ()
void	gtk_frame_set_shadow_type ()
const gchar *	gtk_frame_get_label ()
void	gtk_frame_get_label_align ()
GtkWidget *	gtk_frame_get_label_widget ()
GtkShadowType	gtk_frame_get_shadow_type ()

Properties

gchar *	label	Read / Write
GtkWidget *	label-widget	Read / Write
gfloat	label-xalign	Read / Write
gfloat	label-yalign	Read / Write
GtkShadowType	shadow-type	Read / Write

Types and Values

struct	GtkFrame
struct	GtkFrameClass

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkContainer
            └── GtkBin
                └── GtkFrame
                    └── GtkAspectFrame
```

Implemented Interfaces

GtkFrame implements AtkImplementorIface and GtkBuildable.

Includes

```
#include <gtk/gtk.h>
```

Description

The frame widget is a bin that surrounds its child with a decorative frame and an optional label. If present, the label is drawn in a gap in the top side of the frame. The position of the label can be controlled with `gtk_frame_set_label_align()`.

GtkFrame as GtkBuildable

The GtkFrame implementation of the GtkBuildable interface supports placing a child in the label position by

specifying “label” as the “type” attribute of a <child> element. A normal content child can be specified without specifying a <child> type attribute.

An example of a UI definition fragment with GtkFrame:

```
<object class="GtkFrame">
  <child type="label">
    <object class="GtkLabel" id="frame-
label"/>
  </child>
  <child>
    <object class="GtkEntry" id="frame-
content"/>
  </child>
</object>
```

CSS nodes

```
frame
├── border[.flat]
├── <label widget>
└── <child>
```

GtkFrame has a main CSS node named “frame” and a subnode named “border”. The “border” node is used to draw the visible border. You can set the appearance of the border using CSS properties like “border-style” on the “border” node.

The border node can be given the style class “.flat”, which is used by themes to disable drawing of the border. To do this from code, call `gtk_frame_set_shadow_type()` with `GTK_SHADOW_NONE` to add the “.flat” class or any other shadow type to remove it.

Functions

gtk_frame_new ()

```
GtkWidget *
gtk_frame_new (const gchar *label);
```

Creates a new GtkFrame, with optional label `label`. If `label` is `NULL`, the label is omitted.

Parameters

label	the text to use as the label of the frame.	[allow-none]

Returns

a new GtkFrame widget

---

gtk_frame_set_label ()

```
void
gtk_frame_set_label (GtkFrame *frame,
                     const gchar *label);
```

Removes the current “label-widget”. If `label` is not `NULL`, creates a new GtkLabel with that text and adds it as the “label-widget”.

Parameters

frame	a GtkFrame	
label	the text to use as the label of the frame.	[allow-none]

---

gtk_frame_set_label_widget ()

```
void
gtk_frame_set_label_widget (GtkFrame *frame,
                            GtkWidget *label_widget);
```

Sets the “label-widget” for the frame. This is the widget that will appear embedded in the top edge of the frame as a title.

Parameters

frame	a GtkFrame	
label_widget	the new label widget.	[nullable]

---

gtk_frame_set_label_align ()

```
void
gtk_frame_set_label_align (GtkFrame *frame,
                           gfloat xalign,
                           gfloat yalign);
```

Sets the alignment of the frame widget’s label. The default values for a newly created frame are 0.0 and 0.5.

Parameters

frame	a GtkFrame
xalign	The position of the label along the top edge of the widget. A value of 0.0 represents left alignment; 1.0 represents right alignment.
yalign	The y alignment of the label. A value of 0.0 aligns under the frame; 1.0 aligns above the frame. If the values are exactly 0.0 or 1.0 the gap in the frame won’t be painted

because the label will be completely above or below the frame.

---

gtk_frame_set_shadow_type ()

```
void
gtk_frame_set_shadow_type (GtkFrame *frame,
                           GtkShadowType type);
```

Sets the “shadow-type” for `frame` , i.e. whether it is drawn without (`GTK_SHADOW_NONE`) or with (other values) a visible border. Values other than `GTK_SHADOW_NONE` are treated identically by GtkFrame. The chosen type is applied by removing or adding the .flat class to the CSS node named border.

Parameters

frame	a GtkFrame
type	the new GtkShadowType

---

gtk_frame_get_label ()

```
const gchar *
gtk_frame_get_label (GtkFrame *frame);
```

If the frame’s label widget is a GtkLabel, returns the text in the label widget. (The frame will have a GtkLabel for the label widget if a non-`NULL` argument was passed to `gtk_frame_new()`.)

Parameters

frame	a GtkFrame

Returns

the text in the label, or `NULL` if there was no label widget or the lable widget was not a GtkLabel. This string is owned by GTK+ and must not be modified or freed.

[nullable]

---

gtk_frame_get_label_align ()

```
void
gtk_frame_get_label_align (GtkFrame *frame,
                           gfloat *xalign,
                           gfloat *yalign);
```

Retrieves the X and Y alignment of the frame’s label. See `gtk_frame_set_label_align()`.

Parameters

frame	a GtkFrame	
xalign	location to store X alignment of frame's label, or NULL.	[out][allow-none]
yalign	location to store X alignment of frame's label, or NULL.	[out][allow-none]

---

gtk_frame_get_label_widget ()

```
GtkWidget *
gtk_frame_get_label_widget (GtkFrame *frame);
```

Retrieves the label widget for the frame. See gtk_frame_set_label_widget().

Parameters

frame	a GtkFrame

Returns

the label widget, or NULL if there is none.

[nullable][transfer none]

---

gtk_frame_get_shadow_type ()

```
GtkShadowType
gtk_frame_get_shadow_type (GtkFrame *frame);
```

Retrieves the shadow type of the frame. See gtk_frame_set_shadow_type().

Parameters

frame	a GtkFrame

Returns

the current shadow type of the frame.

Types and Values

struct GtkFrame

```
struct GtkFrame;
```

---

struct GtkFrameClass

```
struct GtkFrameClass {
  GtkBinClass parent_class;


  void (*compute_child_allocation) (GtkFrame *frame,
                                    GtkAllocation *allocation);
};
```

Members

`compute_child_allocation` ()

Property Details

The `"label"` property

`"label"` `gchar *`

Text of the frame's label.

Flags: Read / Write

Default value: NULL

The `"label-widget"` property

`"label-widget"` `GtkWidget *`

A widget to display in place of the usual frame label.

Flags: Read / Write

The `"label-xalign"` property

`"label-xalign"` `gfloat`

The horizontal alignment of the label.

Flags: Read / Write

Allowed values: [0,1]

Default value: 0

The `"label-yalign"` property

`"label-yalign"` `gfloat`

The vertical alignment of the label.

Flags: Read / Write

Allowed values: [0,1]

Default value: 0.5

The `"shadow-type"` property

`"shadow-type"` `GtkShadowType`

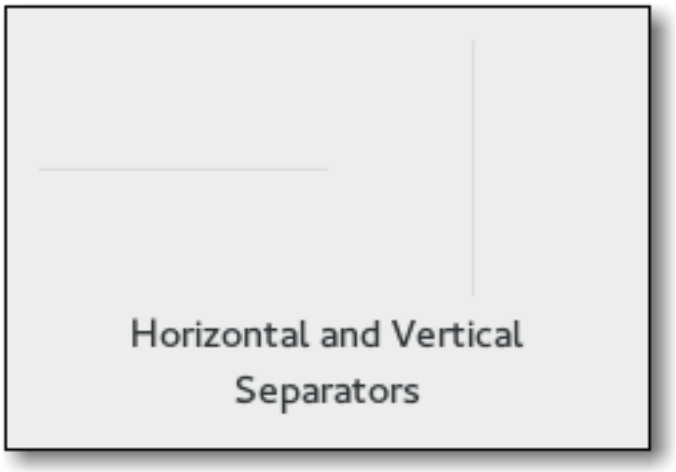
Appearance of the frame border.

Flags: Read / Write

Default value: GTK_SHADOW_ETCHED_IN

GtkSeparator

GtkSeparator — A separator widget



Functions

GtkWidget *	gtk_separator_new ()

Types and Values

struct	GtkSeparator

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkSeparator
            ├── GtkHSeparator
            └── GtkVSeparator
```

Implemented Interfaces

GtkSeparator implements AtkImplementorIface, GtkBuildable and GtkOrientable.

Includes

```
#include <gtk/gtk.h>
```

Description

GtkSeparator is a horizontal or vertical separator widget, depending on the value of the “orientation” property, used to group the widgets within a window. It displays a line with a shadow to make it appear sunken into the interface.

CSS nodes

GtkSeparator has a single CSS node with name separator. The node gets one of the .horizontal or .vertical style classes.

Functions

gtk_separator_new ()

```
GtkWidget *
gtk_separator_new (GtkOrientation orientation);
```

Creates a new GtkSeparator with the given orientation.

Parameters

orientation	the separator’s orientation.

Returns

a new GtkSeparator.

Since: 3.0

Types and Values

struct GtkSeparator

```
struct GtkSeparator;
```

Scrolling

GtkScrollbar — A Scrollbar

GtkScrolledWindow — Adds scrollbars to its child widget

GtkScrollable — An interface for scrollable widgets

GtkScrollbar

GtkScrollbar — A Scrollbar

Functions

GtkWidget *	gtk_scrollbar_new ()

Style Properties

gboolean	fixed-slider-length	Read
gboolean	has-backward-stepper	Read
gboolean	has-forward-stepper	Read
gboolean	has-secondary-backward-stepper	Read
gboolean	has-secondary-forward-stepper	Read
gint	min-slider-length	Read

Types and Values

struct	GtkScrollbar

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkRange
            └── GtkScrollbar
                ├── GtkHScrollbar
                └── GtkVScrollbar
```

Implemented Interfaces

GtkScrollbar implements AtkImplementorIface, GtkBuildable and GtkOrientable.

Includes

```
#include <gtk/gtk.h>
```

Description

The GtkScrollbar widget is a horizontal or vertical scrollbar, depending on the value of the “orientation” property.

Its position and movement are controlled by the adjustment that is passed to or created by `gtk_scrollbar_new()`. See GtkAdjustment for more details. The “value” field sets the position of the thumb and must be between “lower” and “upper” - “page-size”. The “page-size” represents the size of the visible scrollable area. The fields “step-increment” and “page-increment” fields are added to or subtracted from the “value” when the user asks to move by a step (using e.g. the cursor arrow keys or, if present, the stepper buttons) or by a page (using e.g. the Page Down/Up keys).

CSS nodes

```
scrollbar[.fine-tune]
└── contents
    ├── [button.up]
    ├── [button.down]
    ├── trough
    │   └── slider
    ├── [button.up]
    └── [button.down]
```

GtkScrollbar has a main CSS node with name scrollbar and a subnode for its contents, with subnodes named trough and slider.

The main node gets the style class .fine-tune added when the scrollbar is in 'fine-tuning' mode.

If steppers are enabled, they are represented by up to four additional subnodes with name button. These get the style classes .up and .down to indicate in which direction they are moving.

Other style classes that may be added to scrollbars inside GtkScrolledWindow include the positional classes (.left, .right, .top, .bottom) and style classes related to overlay scrolling (.overlay-indicator, .dragging, .hovering).

Functions

gtk_scrollbar_new ()

```
GtkWidget *
gtk_scrollbar_new (GtkOrientation orientation,
                   GtkAdjustment *adjustment);
```

Creates a new scrollbar with the given orientation.

Parameters

orientation	the scrollbar’s orientation.	
adjustment	the GtkAdjustment to use, or `NULL` to create a new adjustment.	[allow-none]

Returns

the new GtkScrollbar.

Since: 3.0

Types and Values

struct GtkScrollbar

```
struct GtkScrollbar;
```

Style Property Details

The “fixed-slider-length” style property

```
  “fixed-slider-length”      gboolean
```

Don't change slider size, just lock it to the minimum length.

Flags: Read

Default value: FALSE

The “has-backward-stepper” style property

```
  “has-backward-stepper”      gboolean
```

Display the standard backward arrow button.

Flags: Read

Default value: TRUE

The “has-forward-stepper” style property

```
  “has-forward-stepper”      gboolean
```

Display the standard forward arrow button.

Flags: Read

Default value: TRUE

The “has-secondary-backward-stepper” style property

```
  “has-secondary-backward-stepper” gboolean
```

Display a second backward arrow button on the opposite end of the scrollbar.

Flags: Read

Default value: FALSE

The “has-secondary-forward-stepper” style property

“has-secondary-forward-stepper” gboolean

Display a second forward arrow button on the opposite end of the scrollbar.

Flags: Read

Default value: FALSE

The “min-slider-length” style property

“min-slider-length” gint

Minimum length of scrollbar slider.

GtkScrollbar:min-slider-length has been deprecated since version 3.20 and should not be used in newly-written code.

Use min-height/min-width CSS properties on the slider element instead. The value of this style property is ignored.

Flags: Read

Allowed values: >= 0

Default value: 21

See Also

GtkAdjustment, GtkScrolledWindow

GtkScrolledWindow

GtkScrolledWindow — Adds scrollbars to its child widget

Functions

GtkWidget *	gtk_scrolled_window_new ()
GtkAdjustment *	gtk_scrolled_window_get_hadjustment ()
void	gtk_scrolled_window_set_hadjustment ()
GtkAdjustment *	gtk_scrolled_window_get_vadjustment ()
void	gtk_scrolled_window_set_vadjustment ()
GtkWidget *	gtk_scrolled_window_get_hscrollbar ()
GtkWidget *	gtk_scrolled_window_get_vscrollbar ()
void	gtk_scrolled_window_get_policy ()
void	gtk_scrolled_window_set_policy ()
void	gtk_scrolled_window_add_with_viewport ()
GtkCornerType	gtk_scrolled_window_get_placement ()
void	gtk_scrolled_window_set_placement ()
void	gtk_scrolled_window_unset_placement ()
GtkShadowType	gtk_scrolled_window_get_shadow_type ()
void	gtk_scrolled_window_set_shadow_type ()
gboolean	gtk_scrolled_window_get_kinetic_scrolling ()
void	gtk_scrolled_window_set_kinetic_scrolling ()
gboolean	gtk_scrolled_window_get_capture_button_press ()
void	gtk_scrolled_window_set_capture_button_press ()
gboolean	gtk_scrolled_window_get_overlay_scrolling ()
void	gtk_scrolled_window_set_overlay_scrolling ()
gint	gtk_scrolled_window_get_min_content_width ()
void	gtk_scrolled_window_set_min_content_width ()
gint	gtk_scrolled_window_get_min_content_height ()
void	gtk_scrolled_window_set_min_content_height ()
gint	gtk_scrolled_window_get_max_content_width ()
void	gtk_scrolled_window_set_max_content_width ()
gint	gtk_scrolled_window_get_max_content_height ()
void	gtk_scrolled_window_set_max_content_height ()
gboolean	gtk_scrolled_window_get_propagate_natural_width ()
void	gtk_scrolled_window_set_propagate_natural_width ()
gboolean	gtk_scrolled_window_get_propagate_natural_height ()
void	gtk_scrolled_window_set_propagate_natural_height ()

Properties

GtkAdjustment *	hadjustment	Read / Write / Construct
GtkPolicyType	hscrollbar-policy	Read / Write
gboolean	kinetic-scrolling	Read / Write
gint	max-content-height	Read / Write
gint	max-content-width	Read / Write
gint	min-content-height	Read / Write
gint	min-content-width	Read / Write
gboolean	overlay-scrolling	Read / Write
gboolean	propagate-natural-height	Read / Write
gboolean	propagate-natural-width	Read / Write
GtkShadowType	shadow-type	Read / Write
GtkAdjustment *	vadjustment	Read / Write / Construct
GtkPolicyType	vscrollbar-policy	Read / Write
GtkCornerType	window-placement	Read / Write

gboolean	window-placement-set	Read / Write

Style Properties

gint	scrollbar-spacing	Read
gboolean	scrollbars-within-bevel	Read

Signals

void	edge-overshot	Run Last
void	edge-reached	Run Last
void	move-focus-out	Action
gboolean	scroll-child	Action

Types and Values

struct	GtkScrolledWindow
struct	GtkScrolledWindowClass
enum	GtkPolicyType
enum	GtkCornerType

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkContainer
            └── GtkBin
                └── GtkScrolledWindow
                    └── GtkPlacesSidebar
```

Implemented Interfaces

GtkScrolledWindow implements AtkImplementorIface and GtkBuildable.

Includes

```
#include <gtk/gtk.h>
```

Description

GtkScrolledWindow is a container that accepts a single child widget, makes that child scrollable using either internally added scrollbars or externally associated adjustments, and optionally draws a frame around the child.

Widgets with native scrolling support, i.e. those whose classes implement the GtkScrollable interface, are added directly. For other types of widget, the class GtkViewport acts as an adaptor, giving scrollability to other widgets. GtkScrolledWindow's implementation of `gtk_container_add()` intelligently accounts for whether or not the added child is a GtkScrollable. If it isn't, GtkScrolledWindow wraps the child in a GtkViewport and adds that for you. Therefore, you can just add any child widget and not worry about the details.

If `gtk_container_add()` has added a GtkViewport for you, you can remove both your added child widget from the GtkViewport, and the GtkViewport from the GtkScrolledWindow, like this:

```
GtkWidget *scrolled_window =
gtk_scrolled_window_new (NULL, NULL);
GtkWidget *child_widget = gtk_button_new ();

// GtkButton is not a GtkScrollable, so
GtkScrolledWindow will automatically
// add a GtkViewport.
gtk_container_add (GTK_CONTAINER
(scrolled_window),
                   child_widget);

// Either of these will result in
child_widget being unparented:
gtk_container_remove (GTK_CONTAINER
(scrolled_window),
                      child_widget);
// or
gtk_container_remove (GTK_CONTAINER
(scrolled_window),
                      gtk_bin_get_child
(GTK_BIN (scrolled_window)));
```

Unless “policy” is GTK_POLICY_NEVER or GTK_POLICY_EXTERNAL, GtkScrolledWindow adds internal GtkScrollbar widgets around its child. The scroll position of the child, and if applicable the scrollbars, is controlled by the “hadjustment” and “vadjustment” that are associated with the GtkScrolledWindow. See the docs on GtkScrollbar for the details, but note that the “step_increment” and “page_increment” fields are only effective if the policy causes scrollbars to be present.

If a GtkScrolledWindow doesn’t behave quite as you would like, or doesn’t have exactly the right layout, it’s very possible to set up your own scrolling with GtkScrollbar and for example a GtkGrid.

Touch support

GtkScrolledWindow has built-in support for touch devices. When a touchscreen is used, swiping will move the scrolled window, and will expose 'kinetic' behavior. This can be turned off with the “kinetic-scrolling” property if it is undesired.

GtkScrolledWindow also displays visual 'overshoot' indication when the content is pulled beyond the end, and this situation can be captured with the “edge-overshot” signal.

If no mouse device is present, the scrollbars will overlayed as narrow, auto-hiding indicators over the content. If traditional scrollbars are desired although no mouse is present, this behaviour can be turned off with the “overlay-scrolling” property.

CSS nodes

GtkScrolledWindow has a main CSS node with name scrolledwindow.

It uses subnodes with names overshoot and undershoot to draw the overflow and underflow indications. These nodes get the .left, .right, .top or .bottom style class added depending on where the indication is drawn.

GtkScrolledWindow also sets the positional style classes (.left, .right, .top, .bottom) and style classes related to overlay scrolling (.overlay-indicator, .dragging, .hovering) on its scrollbars.

If both scrollbars are visible, the area where they meet is drawn with a subnode named junction.

Functions

gtk_scrolled_window_new ()

```
GtkWidget *
gtk_scrolled_window_new (GtkAdjustment *hadjustment,
                         GtkAdjustment *vadjustment);
```

Creates a new scrolled window.

The two arguments are the scrolled window's adjustments; these will be shared with the scrollbars and the child widget to keep the bars in sync with the child. Usually you want to pass `NULL` for the adjustments, which will cause the scrolled window to create them for you.

Parameters

hadjustment	horizontal adjustment.	[nullable]
vadjustment	vertical adjustment.	[nullable]

Returns

a new scrolled window

---

gtk_scrolled_window_get_hadjustment ()

```
GtkAdjustment *
gtk_scrolled_window_get_hadjustment (GtkScrolledWindow *scrolled_window);
```

Returns the horizontal scrollbar's adjustment, used to connect the horizontal scrollbar to the child widget's horizontal scroll functionality.

Parameters

scrolled_window	a GtkScrolledWindow

Returns

the horizontal GtkAdjustment.

[transfer none]

---

gtk_scrolled_window_set_hadjustment ()

```
void
```

```
gtk_scrolled_window_set_hadjustment (GtkScrolledWindow *scrolled_window,
                                     GtkAdjustment *hadjustment);
```
Sets the GtkAdjustment for the horizontal scrollbar.

Parameters

scrolled_window	a GtkScrolledWindow	
hadjustment	the GtkAdjustment to use, or `NULL` to create a new one.	[nullable]

---

gtk_scrolled_window_get_vadjustment ()

```
GtkAdjustment *
gtk_scrolled_window_get_vadjustment (GtkScrolledWindow *scrolled_window);
```
Returns the vertical scrollbar's adjustment, used to connect the vertical scrollbar to the child widget's vertical scroll functionality.

Parameters

scrolled_window	a GtkScrolledWindow

Returns

the vertical GtkAdjustment.

[transfer none]

---

gtk_scrolled_window_set_vadjustment ()

```
void
gtk_scrolled_window_set_vadjustment (GtkScrolledWindow *scrolled_window,
                                     GtkAdjustment *vadjustment);
```
Sets the GtkAdjustment for the vertical scrollbar.

Parameters

scrolled_window	a GtkScrolledWindow	
vadjustment	the GtkAdjustment to use, or `NULL` to create a new one.	[nullable]

---

gtk_scrolled_window_get_hscrollbar ()

```
GtkWidget *
gtk_scrolled_window_get_hscrollbar (GtkScrolledWindow *scrolled_window);
```

Returns the horizontal scrollbar of `scrolled_window`.

Parameters

scrolled_window	a GtkScrolledWindow	

Returns

the horizontal scrollbar of the scrolled window.

[transfer none]

Since: 2.8

---

gtk_scrolled_window_get_vscrollbar ()

```
GtkWidget *
gtk_scrolled_window_get_vscrollbar (GtkScrolledWindow *scrolled_window);
```

Returns the vertical scrollbar of `scrolled_window`.

Parameters

scrolled_window	a GtkScrolledWindow	

Returns

the vertical scrollbar of the scrolled window.

[transfer none]

Since: 2.8

---

gtk_scrolled_window_get_policy ()

```
void
gtk_scrolled_window_get_policy (GtkScrolledWindow *scrolled_window,
                                GtkPolicyType *hscrollbar_policy,
                                GtkPolicyType *vscrollbar_policy);
```

Retrieves the current policy values for the horizontal and vertical scrollbars. See `gtk_scrolled_window_set_policy()`.

Parameters

scrolled_window	a GtkScrolledWindow	
hscrollbar_policy	location to store the policy for the horizontal scrollbar, or `NULL`.	[out][optional]
vscrollbar_policy	location to store the policy for the	[out][optional]

vertical scrollbar, or NULL.

gtk_scrolled_window_set_policy ()

```
void
gtk_scrolled_window_set_policy (GtkScrolledWindow *scrolled_window,
                                GtkPolicyType hscrollbar_policy,
                                GtkPolicyType vscrollbar_policy);
```

Sets the scrollbar policy for the horizontal and vertical scrollbars.

The policy determines when the scrollbar should appear; it is a value from the GtkPolicyType enumeration. If GTK_POLICY_ALWAYS, the scrollbar is always present; if GTK_POLICY_NEVER, the scrollbar is never present; if GTK_POLICY_AUTOMATIC, the scrollbar is present only if needed (that is, if the slider part of the bar would be smaller than the trough — the display is larger than the page size).

Parameters

scrolled_window	a GtkScrolledWindow
hscrollbar_policy	policy for horizontal bar
vscrollbar_policy	policy for vertical bar

gtk_scrolled_window_add_with_viewport ()

```
void
gtk_scrolled_window_add_with_viewport (GtkScrolledWindow *scrolled_window,
                                       GtkWidget *child);
```

`gtk_scrolled_window_add_with_viewport` has been deprecated since version 3.8 and should not be used in newly-written code.

gtk_container_add() will automatically add a GtkViewport if the child doesn't implement GtkScrollable.

Used to add children without native scrolling capabilities. This is simply a convenience function; it is equivalent to adding the unscrollable child to a viewport, then adding the viewport to the scrolled window. If a child has native scrolling, use `gtk_container_add()` instead of this function.

The viewport scrolls the child by moving its GdkWindow, and takes the size of the child to be the size of its toplevel GdkWindow. This will be very wrong for most widgets that support native scrolling; for example, if you add a widget such as GtkTreeView with a viewport, the whole widget will scroll, including the column headings. Thus, widgets with native scrolling support should not be used with the GtkViewport proxy.

A widget supports scrolling natively if it implements the GtkScrollable interface.

Parameters

scrolled_window	a GtkScrolledWindow
child	the widget you want to scroll

gtk_scrolled_window_get_placement ()

```
GtkCornerType
gtk_scrolled_window_get_placement (GtkScrolledWindow *scrolled_window);
```

Gets the placement of the contents with respect to the scrollbars for the scrolled window. See gtk_scrolled_window_set_placement().

Parameters

scrolled_window	a GtkScrolledWindow

Returns

the current placement value.

See also gtk_scrolled_window_set_placement() and gtk_scrolled_window_unset_placement().

---

gtk_scrolled_window_set_placement ()

```
void
gtk_scrolled_window_set_placement (GtkScrolledWindow *scrolled_window,
                                   GtkCornerType window_placement);
```

Sets the placement of the contents with respect to the scrollbars for the scrolled window.

The default is GTK_CORNER_TOP_LEFT, meaning the child is in the top left, with the scrollbars underneath and to the right. Other values in GtkCornerType are GTK_CORNER_TOP_RIGHT, GTK_CORNER_BOTTOM_LEFT, and GTK_CORNER_BOTTOM_RIGHT.

See also gtk_scrolled_window_get_placement() and gtk_scrolled_window_unset_placement().

Parameters

scrolled_window	a GtkScrolledWindow
window_placement	position of the child window

---

gtk_scrolled_window_unset_placement ()

```
void
gtk_scrolled_window_unset_placement (GtkScrolledWindow *scrolled_window);
```

Unsets the placement of the contents with respect to the scrollbars for the scrolled window. If no window placement is set for a scrolled window, it defaults to GTK_CORNER_TOP_LEFT.

See also gtk_scrolled_window_set_placement() and gtk_scrolled_window_get_placement().

Parameters

scrolled_window	a GtkScrolledWindow

Since: 2.10

---

gtk_scrolled_window_get_shadow_type ()

```
GtkShadowType
gtk_scrolled_window_get_shadow_type (GtkScrolledWindow *scrolled_window);
```

Gets the shadow type of the scrolled window. See `gtk_scrolled_window_set_shadow_type()`.

Parameters

scrolled_window	a GtkScrolledWindow

Returns

the current shadow type

---

gtk_scrolled_window_set_shadow_type ()

```
void
gtk_scrolled_window_set_shadow_type (GtkScrolledWindow *scrolled_window,
                                     GtkShadowType type);
```

Changes the type of shadow drawn around the contents of `scrolled_window`.

Parameters

scrolled_window	a GtkScrolledWindow
type	kind of shadow to draw around scrolled window contents

---

gtk_scrolled_window_get_kinetic_scrolling ()

```
gboolean
gtk_scrolled_window_get_kinetic_scrolling
                                (GtkScrolledWindow *scrolled_window);
```

Returns the specified kinetic scrolling behavior.

Parameters

scrolled_window	a GtkScrolledWindow

Returns

the scrolling behavior flags.

Since: 3.4

---

gtk_scrolled_window_set_kinetic_scrolling ()

```
void
gtk_scrolled_window_set_kinetic_scrolling
                                (GtkScrolledWindow *scrolled_window,
                                 gboolean kinetic_scrolling);
```

Turns kinetic scrolling on or off. Kinetic scrolling only applies to devices with source GDK_SOURCE_TOUCHSCREEN.

Parameters

scrolled_window	a GtkScrolledWindow
kinetic_scrolling	TRUE to enable kinetic scrolling

Since: 3.4

---

gtk_scrolled_window_get_capture_button_press ()

```
gboolean
gtk_scrolled_window_get_capture_button_press
                                (GtkScrolledWindow *scrolled_window);
```

Return whether button presses are captured during kinetic scrolling. See gtk_scrolled_window_set_capture_button_press().

Parameters

scrolled_window	a GtkScrolledWindow

Returns

TRUE if button presses are captured during kinetic scrolling

Since: 3.4

---

gtk_scrolled_window_set_capture_button_press ()

```
void
gtk_scrolled_window_set_capture_button_press
                                (GtkScrolledWindow *scrolled_window,
                                 gboolean capture_button_press);
```

Changes the behaviour of scrolled_window with regard to the initial event that possibly starts kinetic scrolling. When capture_button_press is set to TRUE, the event is captured by the scrolled window, and then later replayed if it is meant to go to the child widget.

This should be enabled if any child widgets perform non-reversible actions on "button-press-event". If they

don't, and handle additionally handle “grab-broken-event”, it might be better to set `capture_button_press` to `FALSE`.

This setting only has an effect if kinetic scrolling is enabled.

Parameters

scrolled_window	a GtkScrolledWindow
capture_button_press	`TRUE` to capture button presses

Since: 3.4

---

gtk_scrolled_window_get_overlay_scrolling ()

```
gboolean
gtk_scrolled_window_get_overlay_scrolling
                                (GtkScrolledWindow *scrolled_window);
```

Returns whether overlay scrolling is enabled for this scrolled window.

Parameters

scrolled_window	a GtkScrolledWindow

Returns

`TRUE` if overlay scrolling is enabled

Since: 3.16

---

gtk_scrolled_window_set_overlay_scrolling ()

```
void
gtk_scrolled_window_set_overlay_scrolling
                                (GtkScrolledWindow *scrolled_window,
                                 gboolean overlay_scrolling);
```

Enables or disables overlay scrolling for this scrolled window.

Parameters

scrolled_window	a GtkScrolledWindow
overlay_scrolling	whether to enable overlay scrolling

Since: 3.16

---

gtk_scrolled_window_get_min_content_width ()

```
gint
gtk_scrolled_window_get_min_content_width
                                (GtkScrolledWindow *scrolled_window);
```

Gets the minimum content width of `scrolled_window` , or -1 if not set.

Parameters

scrolled_window	a GtkScrolledWindow

Returns

the minimum content width

Since: 3.0

---

gtk_scrolled_window_set_min_content_width ()

```
void
gtk_scrolled_window_set_min_content_width
                                (GtkScrolledWindow *scrolled_window,
                                 gint width);
```

Sets the minimum width that `scrolled_window` should keep visible. Note that this can and (usually will) be smaller than the minimum size of the content.

It is a programming error to set the minimum content width to a value greater than “max-content-width”.

Parameters

scrolled_window	a GtkScrolledWindow
width	the minimal content width

Since: 3.0

---

gtk_scrolled_window_get_min_content_height ()

```
gint
gtk_scrolled_window_get_min_content_height
                                (GtkScrolledWindow *scrolled_window);
```

Gets the minimal content height of `scrolled_window` , or -1 if not set.

Parameters

scrolled_window	a GtkScrolledWindow

Returns

the minimal content height

Since: 3.0

---

gtk_scrolled_window_set_min_content_height ()

```
void
gtk_scrolled_window_set_min_content_height
                                (GtkScrolledWindow *scrolled_window,
                                 gint height);
```

Sets the minimum height that `scrolled_window` should keep visible. Note that this can and (usually will) be smaller than the minimum size of the content.

It is a programming error to set the minimum content height to a value greater than "max-content-height".

Parameters

scrolled_window	a GtkScrolledWindow
height	the minimal content height

Since: 3.0

---

gtk_scrolled_window_get_max_content_width ()

```
gint
gtk_scrolled_window_get_max_content_width
                                (GtkScrolledWindow *scrolled_window);
```

Returns the maximum content width set.

Parameters

scrolled_window	a GtkScrolledWindow

Returns

the maximum content width, or -1

Since: 3.22

---

gtk_scrolled_window_set_max_content_width ()

```
void
gtk_scrolled_window_set_max_content_width
                                (GtkScrolledWindow *scrolled_window,
                                 gint width);
```

Sets the maximum width that `scrolled_window` should keep visible. The `scrolled_window` will grow up to

this width before it starts scrolling the content.

It is a programming error to set the maximum content width to a value smaller than “min-content-width”.

Parameters

scrolled_window	a GtkScrolledWindow
width	the maximum content width

Since: 3.22

---

gtk_scrolled_window_get_max_content_height ()

```
gint
gtk_scrolled_window_get_max_content_height
                                (GtkScrolledWindow *scrolled_window);
```

Returns the maximum content height set.

Parameters

scrolled_window	a GtkScrolledWindow

Returns

the maximum content height, or -1

Since: 3.22

---

gtk_scrolled_window_set_max_content_height ()

```
void
gtk_scrolled_window_set_max_content_height
                               (GtkScrolledWindow *scrolled_window,
                                gint height);
```

Sets the maximum height that `scrolled_window` should keep visible. The `scrolled_window` will grow up to this height before it starts scrolling the content.

It is a programming error to set the maximum content height to a value smaller than “min-content-height”.

Parameters

scrolled_window	a GtkScrolledWindow
height	the maximum content height

Since: 3.22

---

gtk_scrolled_window_get_propagate_natural_width ()

```
gboolean
gtk_scrolled_window_get_propagate_natural_width
                                (GtkScrolledWindow *scrolled_window);
```

Reports whether the natural width of the child will be calculated and propagated through the scrolled window's requested natural width.

Parameters

scrolled_window	a GtkScrolledWindow

Returns

whether natural width propagation is enabled.

Since: 3.22

---

gtk_scrolled_window_set_propagate_natural_width ()

```
void
gtk_scrolled_window_set_propagate_natural_width
                                (GtkScrolledWindow *scrolled_window,
                                 gboolean propagate);
```

Sets whether the natural width of the child should be calculated and propagated through the scrolled window's requested natural width.

Parameters

scrolled_window	a GtkScrolledWindow
propagate	whether to propagate natural width

Since: 3.22

---

gtk_scrolled_window_get_propagate_natural_height ()

```
gboolean
gtk_scrolled_window_get_propagate_natural_height
                                (GtkScrolledWindow *scrolled_window);
```

Reports whether the natural height of the child will be calculated and propagated through the scrolled window's requested natural height.

Parameters

scrolled_window	a GtkScrolledWindow

Returns

whether natural height propagation is enabled.

Since: 3.22

---

gtk_scrolled_window_set_propagate_natural_height ()

```
void
gtk_scrolled_window_set_propagate_natural_height
                                (GtkScrolledWindow *scrolled_window,
                                 gboolean propagate);
```

Sets whether the natural height of the child should be calculated and propagated through the scrolled window's requested natural height.

Parameters

scrolled_window	a GtkScrolledWindow
propagate	whether to propagate natural height

Since: 3.22

Types and Values

struct GtkScrolledWindow

```
struct GtkScrolledWindow;
```

---

struct GtkScrolledWindowClass

```
struct GtkScrolledWindowClass {
  GtkBinClass parent_class;

  gint scrollbar_spacing;


  /* Action signals for keybindings. Do not connect to these signals
   */

  /* Unfortunately, GtkScrollType is deficient in that there is
   * no horizontal/vertical variants for GTK_SCROLL_START/END,
   * so we have to add an additional boolean flag.
   */
  gboolean (*scroll_child) (GtkScrolledWindow *scrolled_window,
                    GtkScrollType      scroll,
                    gboolean           horizontal);

  void (* move_focus_out) (GtkScrolledWindow *scrolled_window,
                    GtkDirectionType   direction);
};
```

Members

gint `scrollbar_spacing`;	
`scroll_child` ()	Keybinding signal which gets emitted when a keybinding that scrolls is pressed.
`move_focus_out` ()	Keybinding signal which gets emitted when focus is moved away from the scrolled window by a keybinding.

---

enum GtkPolicyType

Determines how the size should be computed to achieve the one of the visibility mode for the scrollbars.

Members

GTK_POLICY_ALWAYS	The scrollbar is always visible. The view size is independent of the content.
GTK_POLICY_AUTOMATIC	The scrollbar will appear and disappear as necessary. For example, when all of a GtkTreeView can not be seen.
GTK_POLICY_NEVER	The scrollbar should never appear. In this mode the content determines the size.
GTK_POLICY_EXTERNAL	Don't show a scrollbar, but don't force the size to follow the content. This can be used e.g. to make multiple scrolled windows share a scrollbar. Since: 3.16

---

enum GtkCornerType

Specifies which corner a child widget should be placed in when packed into a GtkScrolledWindow. This is effectively the opposite of where the scroll bars are placed.

Members

GTK_CORNER_TOP_LEFT	Place the scrollbars on the right and bottom of the widget (default behaviour).
GTK_CORNER_BOTTOM_LEFT	Place the scrollbars on the top and

	right of the widget.
GTK_CORNER_TOP_RIGHT	Place the scrollbars on the left and bottom of the widget.
GTK_CORNER_BOTTOM_RIGHT	Place the scrollbars on the top and left of the widget.

Property Details

The “hadjustment” property

`“hadjustment” GtkAdjustment *`

The GtkAdjustment for the horizontal position.

Flags: Read / Write / Construct

The “hscrollbar-policy” property

`“hscrollbar-policy” GtkPolicyType`

When the horizontal scrollbar is displayed.

Flags: Read / Write

Default value: GTK_POLICY_AUTOMATIC

The “kinetic-scrolling” property

`“kinetic-scrolling” gboolean`

Whether kinetic scrolling is enabled or not. Kinetic scrolling only applies to devices with source `GDK_SOURCE_TOUCHSCREEN`.

Flags: Read / Write

Default value: TRUE

Since: 3.4

The “max-content-height” property

`“max-content-height” gint`

The maximum content height of `scrolled_window` , or -1 if not set.

Flags: Read / Write

Allowed values: >= -1

Default value: -1

Since: 3.22

The “max-content-width” property

“max-content-width” gint

The maximum content width of scrolled_window , or -1 if not set.

Flags: Read / Write

Allowed values: >= -1

Default value: -1

Since: 3.22

The “min-content-height” property

“min-content-height” gint

The minimum content height of scrolled_window , or -1 if not set.

Flags: Read / Write

Allowed values: >= -1

Default value: -1

Since: 3.0

The “min-content-width” property

“min-content-width” gint

The minimum content width of scrolled_window , or -1 if not set.

Flags: Read / Write

Allowed values: >= -1

Default value: -1

Since: 3.0

The “overlay-scrolling” property

“overlay-scrolling” gboolean

Whether overlay scrolling is enabled or not. If it is, the scrollbars are only added as traditional widgets when a mouse is present. Otherwise, they are overlayed on top of the content, as narrow indicators.

Note that overlay scrolling can also be globally disabled, with the “gtk-overlay-scrolling” setting.

Flags: Read / Write

Default value: TRUE

Since: 3.16

The “propagate-natural-height” property

“propagate-natural-height” gboolean

Whether the natural height of the child should be calculated and propagated through the scrolled window’s requested natural height.

This is useful in cases where an attempt should be made to allocate exactly enough space for the natural size of the child.

Flags: Read / Write

Default value: FALSE

Since: 3.22

The “propagate-natural-width” property

“propagate-natural-width” gboolean

Whether the natural width of the child should be calculated and propagated through the scrolled window’s requested natural width.

This is useful in cases where an attempt should be made to allocate exactly enough space for the natural size of the child.

Flags: Read / Write

Default value: FALSE

Since: 3.22

The “shadow-type” property

“shadow-type” GtkShadowType

Style of bevel around the contents.

Flags: Read / Write

Default value: GTK_SHADOW_NONE

The “vadjustment” property

“vadjustment” GtkAdjustment *

The GtkAdjustment for the vertical position.

Flags: Read / Write / Construct

The “vscrollbar-policy” property

“vscrollbar-policy” GtkPolicyType

When the vertical scrollbar is displayed.

Flags: Read / Write

Default value: GTK_POLICY_AUTOMATIC

The “window-placement” property

“window-placement” GtkCornerType

Where the contents are located with respect to the scrollbars.

Flags: Read / Write

Default value: GTK_CORNER_TOP_LEFT

The “window-placement-set” property

“window-placement-set” gboolean

Whether "window-placement" should be used to determine the location of the contents with respect to the scrollbars.

GtkScrolledWindow:window-placement-set has been deprecated since version 3.10 and should not be used in newly-written code.

This value is ignored and “window-placement” value is always honored.

Flags: Read / Write

Default value: TRUE

Since: 2.10

Style Property Details

The “scrollbar-spacing” style property

“scrollbar-spacing” gint

Number of pixels between the scrollbars and the scrolled window.

Flags: Read

Allowed values: >= 0

Default value: 3

The “scrollbars-within-bevel” style property

```
  “scrollbars-within-bevel”  gboolean
```

Whether to place scrollbars within the scrolled window's bevel.

`GtkScrolledWindow:scrollbars-within-bevel` has been deprecated since version 3.20 and should not be used in newly-written code.

the value of this style property is ignored.

Flags: Read

Default value: FALSE

Since: 2.12

Signal Details

The “edge-overshot” signal

```
void
user_function (GtkScrolledWindow *scrolled_window,
               GtkPositionType    pos,
               gpointer           user_data)
```

The ::edge-overshot signal is emitted whenever user initiated scrolling makes the scrolled window firmly surpass (i.e. with some edge resistance) the lower or upper limits defined by the adjustment in that orientation.

A similar behavior without edge resistance is provided by the “edge-reached” signal.

Note: The `pos` argument is LTR/RTL aware, so callers should be aware too if intending to provide behavior on horizontal edges.

Parameters

scrolled_window	a GtkScrolledWindow
pos	edge side that was hit
user_data	user data set when the signal handler was connected.

Flags: Run Last

Since: 3.16

The “edge-reached” signal

```
void
user_function (GtkScrolledWindow *scrolled_window,
               GtkPositionType    pos,
               gpointer           user_data)
```

The ::edge-reached signal is emitted whenever user-initiated scrolling makes the scrolled window exactly reach the lower or upper limits defined by the adjustment in that orientation.

A similar behavior with edge resistance is provided by the “edge-overshot” signal.

Note: The `pos` argument is LTR/RTL aware, so callers should be aware too if intending to provide behavior on horizontal edges.

Parameters

scrolled_window	a GtkScrolledWindow
pos	edge side that was reached
user_data	user data set when the signal handler was connected.

Flags: Run Last

Since: 3.16

The “move-focus-out” signal

```
void
user_function (GtkScrolledWindow *scrolled_window,
               GtkDirectionType   direction_type,
               gpointer           user_data)
```

The ::move-focus-out signal is a keybinding signal which gets emitted when focus is moved away from the scrolled window by a keybinding. The “move-focus” signal is emitted with `direction_type` on this scrolled window’s toplevel parent in the container hierarchy. The default bindings for this signal are `Ctrl + Tab` to move forward and `Ctrl + Shift + Tab` to move backward.

Parameters

scrolled_window	a GtkScrolledWindow
direction_type	either `GTK_DIR_TAB_FORWARD` or `GTK_DIR_TAB_BACKWARD`
user_data	user data set when the signal handler was connected.

Flags: Action

The “scroll-child” signal

```
gboolean
user_function (GtkScrolledWindow *scrolled_window,
               GtkScrollType      scroll,
               gboolean           horizontal,
               gpointer           user_data)
```

The ::scroll-child signal is a keybinding signal which gets emitted when a keybinding that scrolls is pressed. The horizontal or vertical adjustment is updated which triggers a signal that the scrolled window’s child may listen to and scroll itself.

Parameters

scrolled_window	a GtkScrolledWindow
scroll	a GtkScrollType describing how much to scroll
horizontal	whether the keybinding scrolls the child horizontally or not
user_data	user data set when the signal handler was connected.

Flags: Action

See Also

GtkScrollable, GtkViewport, GtkAdjustment

GtkScrollable

GtkScrollable — An interface for scrollable widgets

Functions

GtkAdjustment *	gtk_scrollable_get_hadjustment ()
void	gtk_scrollable_set_hadjustment ()
GtkAdjustment *	gtk_scrollable_get_vadjustment ()
void	gtk_scrollable_set_vadjustment ()
GtkScrollablePolicy	gtk_scrollable_get_hscroll_policy ()
void	gtk_scrollable_set_hscroll_policy ()
GtkScrollablePolicy	gtk_scrollable_get_vscroll_policy ()
void	gtk_scrollable_set_vscroll_policy ()
gboolean	gtk_scrollable_get_border ()

Properties

GtkAdjustment *	hadjustment	Read / Write / Construct
GtkScrollablePolicy	hscroll-policy	Read / Write
GtkAdjustment *	vadjustment	Read / Write / Construct
GtkScrollablePolicy	vscroll-policy	Read / Write

Types and Values

	GtkScrollable
enum	GtkScrollablePolicy

Object Hierarchy

```
GInterface
└── GtkScrollable
```

Prerequisites

GtkScrollable requires GObject.

Known Implementations

GtkScrollable is implemented by GtkIconView, GtkLayout, GtkTextView, GtkToolPalette, GtkTreeView and GtkViewport.

Includes

```
#include <gtk/gtk.h>
```

Description

GtkScrollable is an interface that is implemented by widgets with native scrolling ability.

To implement this interface you should override the “hadjustment” and “vadjustment” properties.

Creating a scrollable widget

All scrollable widgets should do the following.

- When a parent widget sets the scrollable child widget’s adjustments, the widget should populate the adjustments’ “lower”, “upper”, “step-increment”, “page-increment” and “page-size” properties and connect to the “value-changed” signal.
- Because its preferred size is the size for a fully expanded widget, the scrollable widget must be able to cope with underallocations. This means that it must accept any value passed to its `GtkWidgetClass.size_allocate()` function.
- When the parent allocates space to the scrollable child widget, the widget should update the adjustments’ properties with new values.
- When any of the adjustments emits the “value-changed” signal, the scrollable widget should scroll its contents.

Functions

gtk_scrollable_get_hadjustment ()

```
GtkAdjustment *
gtk_scrollable_get_hadjustment (GtkScrollable *scrollable);
```

Retrieves the GtkAdjustment used for horizontal scrolling.

Parameters

scrollable	a GtkScrollable	

Returns

horizontal GtkAdjustment.

[transfer none]

Since: 3.0

---

gtk_scrollable_set_hadjustment ()

```
void
gtk_scrollable_set_hadjustment (GtkScrollable *scrollable,
                                GtkAdjustment *hadjustment);
```

Sets the horizontal adjustment of the GtkScrollable.

Parameters

scrollable	a GtkScrollable	
hadjustment	a GtkAdjustment.	[allow-none]

Since: 3.0

---

gtk_scrollable_get_vadjustment ()

```
GtkAdjustment *
gtk_scrollable_get_vadjustment (GtkScrollable *scrollable);
```

Retrieves the GtkAdjustment used for vertical scrolling.

Parameters

scrollable	a GtkScrollable	

Returns

vertical GtkAdjustment.

[transfer none]

Since: 3.0

---

gtk_scrollable_set_vadjustment ()

```
void
gtk_scrollable_set_vadjustment (GtkScrollable *scrollable,
                                GtkAdjustment *vadjustment);
```

Sets the vertical adjustment of the GtkScrollable.

Parameters

scrollable	a GtkScrollable	
vadjustment	a GtkAdjustment.	[allow-none]

Since: 3.0

---

gtk_scrollable_get_hscroll_policy ()

```
GtkScrollablePolicy
gtk_scrollable_get_hscroll_policy (GtkScrollable *scrollable);
```

Gets the horizontal GtkScrollablePolicy.

Parameters

scrollable	a GtkScrollable

Returns

The horizontal GtkScrollablePolicy.

Since: 3.0

---

gtk_scrollable_set_hscroll_policy ()

```
void
gtk_scrollable_set_hscroll_policy (GtkScrollable *scrollable,
                                   GtkScrollablePolicy policy);
```

Sets the GtkScrollablePolicy to determine whether horizontal scrolling should start below the minimum width or below the natural width.

Parameters

scrollable	a GtkScrollable
policy	the horizontal GtkScrollablePolicy

Since: 3.0

---

gtk_scrollable_get_vscroll_policy ()

```
GtkScrollablePolicy
gtk_scrollable_get_vscroll_policy (GtkScrollable *scrollable);
```

Gets the vertical GtkScrollablePolicy.

Parameters

scrollable	a GtkScrollable	

Returns

The vertical GtkScrollablePolicy.

Since: 3.0

---

gtk_scrollable_set_vscroll_policy ()

```
void
gtk_scrollable_set_vscroll_policy (GtkScrollable *scrollable,
                                   GtkScrollablePolicy policy);
```

Sets the GtkScrollablePolicy to determine whether vertical scrolling should start below the minimum height or below the natural height.

Parameters

scrollable	a GtkScrollable	
policy	the vertical GtkScrollablePolicy	

Since: 3.0

---

gtk_scrollable_get_border ()

```
gboolean
gtk_scrollable_get_border (GtkScrollable *scrollable,
                           GtkBorder *border);
```

Returns the size of a non-scrolling border around the outside of the scrollable. An example for this would be treeview headers. GTK+ can use this information to display overlayed graphics, like the overshoot indication, at the right position.

Parameters

scrollable	a GtkScrollable	
border	return location for the results.	[out caller-allocates]

Returns

`TRUE` if `border` has been set

Since: 3.16

Types and Values

GtkScrollable

```
typedef struct _GtkScrollable GtkScrollable;
```

---

enum GtkScrollablePolicy

Defines the policy to be used in a scrollable widget when updating the scrolled window adjustments in a given orientation.

Members

GTK_SCROLL_MINIMUM	Scrollable adjustments are based on the minimum size
GTK_SCROLL_NATURAL	Scrollable adjustments are based on the natural size

Property Details

The `"hadjustment"` property

```
  "hadjustment"              GtkAdjustment *
```

Horizontal GtkAdjustment of the scrollable widget. This adjustment is shared between the scrollable widget and its parent.

Flags: Read / Write / Construct

Since: 3.0

The `"hscroll-policy"` property

```
  "hscroll-policy"           GtkScrollablePolicy
```

Determines whether horizontal scrolling should start once the scrollable widget is allocated less than its minimum width or less than its natural width.

Flags: Read / Write

Default value: GTK_SCROLL_MINIMUM

Since: 3.0

The “vadjustment” property

“vadjustment” GtkAdjustment *

Verical GtkAdjustment of the scrollable widget. This adjustment is shared between the scrollable widget and its parent.

Flags: Read / Write / Construct

Since: 3.0

The “vscroll-policy” property

“vscroll-policy” GtkScrollablePolicy

Determines whether vertical scrolling should start once the scrollable widget is allocated less than its minimum height or less than its natural height.

Flags: Read / Write

Default value: GTK_SCROLL_MINIMUM

Since: 3.0

Printing

GtkPrintOperation — High-level Printing API

GtkPrintContext — Encapsulates context for drawing pages

GtkPrintSettings — Stores print settings

GtkPageSetup — Stores page setup information

GtkPaperSize — Support for named paper sizes

GtkPrinter — Represents a printer

GtkPrintJob — Represents a print job

GtkPrintUnixDialog — A print dialog

GtkPageSetupUnixDialog — A page setup dialog

GtkPrintOperation

GtkPrintOperation — High-level Printing API

Functions

GtkPrintOperation *	gtk_print_operation_new ()
void	gtk_print_operation_set_allow_async ()
void	gtk_print_operation_get_error ()
void	gtk_print_operation_set_default_page_setup ()
GtkPageSetup *	gtk_print_operation_get_default_page_setup ()
void	gtk_print_operation_set_print_settings ()
GtkPrintSettings *	gtk_print_operation_get_print_settings ()
void	gtk_print_operation_set_job_name ()
void	gtk_print_operation_set_n_pages ()
gint	gtk_print_operation_get_n_pages_to_print ()
void	gtk_print_operation_set_current_page ()
void	gtk_print_operation_set_use_full_page ()
void	gtk_print_operation_set_unit ()
void	gtk_print_operation_set_export_filename ()
void	gtk_print_operation_set_show_progress ()
void	gtk_print_operation_set_track_print_status ()
void	gtk_print_operation_set_custom_tab_label ()
GtkPrintOperationResult	gtk_print_operation_run ()
void	gtk_print_operation_cancel ()
void	gtk_print_operation_draw_page_finish ()
void	gtk_print_operation_set_defer_drawing ()
GtkPrintStatus	gtk_print_operation_get_status ()
const gchar *	gtk_print_operation_get_status_string ()
gboolean	gtk_print_operation_is_finished ()
void	gtk_print_operation_set_support_selection ()
gboolean	gtk_print_operation_get_support_selection ()
void	gtk_print_operation_set_has_selection ()
gboolean	gtk_print_operation_get_has_selection ()
void	gtk_print_operation_set_embed_page_setup ()
gboolean	gtk_print_operation_get_embed_page_setup ()
GtkPageSetup *	gtk_print_run_page_setup_dialog ()
void	(*GtkPageSetupDoneFunc) ()
void	gtk_print_run_page_setup_dialog_async ()
void	gtk_print_operation_preview_end_preview ()
gboolean	gtk_print_operation_preview_is_selected ()
void	gtk_print_operation_preview_render_page ()

Properties

gboolean	allow-async	Read / Write
gint	current-page	Read / Write
gchar *	custom-tab-label	Read / Write
GtkPageSetup *	default-page-setup	Read / Write
gboolean	embed-page-setup	Read / Write
gchar *	export-filename	Read / Write
gboolean	has-selection	Read / Write
gchar *	job-name	Read / Write
gint	n-pages	Read / Write
gint	n-pages-to-print	Read
GtkPrintSettings *	print-settings	Read / Write

gboolean	show-progress	Read / Write
GtkPrintStatus	status	Read
gchar *	status-string	Read
gboolean	support-selection	Read / Write
gboolean	track-print-status	Read / Write
GtkUnit	unit	Read / Write
gboolean	use-full-page	Read / Write

Signals

void	begin-print	Run Last
GObject*	create-custom-widget	Run Last
void	custom-widget-apply	Run Last
void	done	Run Last
void	draw-page	Run Last
void	end-print	Run Last
gboolean	paginate	Run Last
gboolean	preview	Run Last
void	request-page-setup	Run Last
void	status-changed	Run Last
void	update-custom-widget	Run Last
void	got-page-size	Run Last
void	ready	Run Last

Types and Values

struct	GtkPrintOperation
struct	GtkPrintOperationClass
enum	GtkPrintStatus
enum	GtkPrintOperationAction
enum	GtkPrintOperationResult
enum	GtkPrintError
#define	GTK_PRINT_ERROR
	GtkPrintOperationPreview

Object Hierarchy

```
GInterface
└── GtkPrintOperationPreview
GObject
└── GtkPrintOperation
```

Prerequisites

GtkPrintOperationPreview requires GObject.

Implemented Interfaces

GtkPrintOperation implements GtkPrintOperationPreview.

Known Implementations

GtkPrintOperationPreview is implemented by GtkPrintOperation.

Includes

```
#include <gtk/gtk.h>
```

Description

GtkPrintOperation is the high-level, portable printing API. It looks a bit different than other GTK+ dialogs such as the GtkFileChooser, since some platforms don't expose enough infrastructure to implement a good print dialog. On such platforms, GtkPrintOperation uses the native print dialog. On platforms which do not provide a native print dialog, GTK+ uses its own, see GtkPrintUnixDialog.

The typical way to use the high-level printing API is to create a GtkPrintOperation object with `gtk_print_operation_new()` when the user selects to print. Then you set some properties on it, e.g. the page size, any GtkPrintSettings from previous print operations, the number of pages, the current page, etc.

Then you start the print operation by calling `gtk_print_operation_run()`. It will then show a dialog, let the user select a printer and options. When the user finished the dialog various signals will be emitted on the GtkPrintOperation, the main one being "draw-page", which you are supposed to catch and render the page on the provided GtkPrintContext using Cairo.

The high-level printing API

```
static GtkPrintSettings *settings = NULL;

static void
do_print (void)
{
  GtkPrintOperation *print;
  GtkPrintOperationResult res;

  print = gtk_print_operation_new ();

  if (settings != NULL)
    gtk_print_operation_set_print_settings (print, settings);

  g_signal_connect (print, "begin_print", G_CALLBACK (begin_print), NULL);
  g_signal_connect (print, "draw_page", G_CALLBACK (draw_page), NULL);

  res = gtk_print_operation_run (print, GTK_PRINT_OPERATION_ACTION_PRINT_DIALOG,
                                 GTK_WINDOW (main_window), NULL);
```

```
24
25                                                  if (res ==
26                                                GTK_PRINT_OPERATION_RESULT_APPLY)
27                                                    {
28                                                      if (settings != NULL)
                                                          g_object_unref (settings);
                                                      settings = g_object_ref
                                                (gtk_print_operation_get_print_settings
                                                (print));
                                                    }

                                                  g_object_unref (print);
                                                }
```

By default GtkPrintOperation uses an external application to do print preview. To implement a custom print preview, an application must connect to the preview signal. The functions `gtk_print_operation_preview_render_page()`, `gtk_print_operation_preview_end_preview()` and `gtk_print_operation_preview_is_selected()` are useful when implementing a print preview.

Functions

gtk_print_operation_new ()

```
GtkPrintOperation *
gtk_print_operation_new (void);
```

Creates a new GtkPrintOperation.

Returns

a new GtkPrintOperation

Since: 2.10

---

gtk_print_operation_set_allow_async ()

```
void
gtk_print_operation_set_allow_async (GtkPrintOperation *op,
                                     gboolean allow_async);
```

Sets whether the `gtk_print_operation_run()` may return before the print operation is completed. Note that some platforms may not allow asynchronous operation.

Parameters

op	a GtkPrintOperation
allow_async	`TRUE` to allow asynchronous operation

Since: 2.10

---

gtk_print_operation_get_error ()

```
void
gtk_print_operation_get_error (GtkPrintOperation *op,
                               GError **error);
```

Call this when the result of a print operation is `GTK_PRINT_OPERATION_RESULT_ERROR`, either as returned by `gtk_print_operation_run()`, or in the "done" signal handler. The returned GError will contain more details on what went wrong.

Parameters

op	a GtkPrintOperation	
error	return location for the error	

Since: 2.10

---

gtk_print_operation_set_default_page_setup ()

```
void
gtk_print_operation_set_default_page_setup
                               (GtkPrintOperation *op,
                                GtkPageSetup *default_page_setup);
```

Makes `default_page_setup` the default page setup for `op` .

This page setup will be used by `gtk_print_operation_run()`, but it can be overridden on a per-page basis by connecting to the "request-page-setup" signal.

Parameters

op	a GtkPrintOperation	
default_page_setup	a GtkPageSetup, or `NULL`.	[allow-none]

Since: 2.10

---

gtk_print_operation_get_default_page_setup ()

```
GtkPageSetup *
gtk_print_operation_get_default_page_setup
                               (GtkPrintOperation *op);
```

Returns the default page setup, see `gtk_print_operation_set_default_page_setup()`.

Parameters

op	a GtkPrintOperation	

Returns

the default page setup.

[transfer none]

Since: 2.10

---

gtk_print_operation_set_print_settings ()

```
void
gtk_print_operation_set_print_settings
                                (GtkPrintOperation *op,
                                 GtkPrintSettings *print_settings);
```

Sets the print settings for `op` . This is typically used to re-establish print settings from a previous print operation, see `gtk_print_operation_run()`.

Parameters

op	a GtkPrintOperation	
print_settings	GtkPrintSettings.	[allow-none]

Since: 2.10

---

gtk_print_operation_get_print_settings ()

```
GtkPrintSettings *
gtk_print_operation_get_print_settings
                                (GtkPrintOperation *op);
```

Returns the current print settings.

Note that the return value is `NULL` until either `gtk_print_operation_set_print_settings()` or `gtk_print_operation_run()` have been called.

Parameters

op	a GtkPrintOperation

Returns

the current print settings of `op` .

[transfer none]

Since: 2.10

---

gtk_print_operation_set_job_name ()

```
void
gtk_print_operation_set_job_name (GtkPrintOperation *op,
                                  const gchar *job_name);
```

Sets the name of the print job. The name is used to identify the job (e.g. in monitoring applications like eggcups).

If you don’t set a job name, GTK+ picks a default one by numbering successive print jobs.

Parameters

op	a GtkPrintOperation
job_name	a string that identifies the print job

Since: 2.10

---

gtk_print_operation_set_n_pages ()

```
void
gtk_print_operation_set_n_pages (GtkPrintOperation *op,
                                 gint n_pages);
```

Sets the number of pages in the document.

This must be set to a positive number before the rendering starts. It may be set in a “begin-print” signal hander.

Note that the page numbers passed to the “request-page-setup” and “draw-page” signals are 0-based, i.e. if the user chooses to print all pages, the last ::draw-page signal will be for page `n_pages` - 1.

Parameters

op	a GtkPrintOperation
n_pages	the number of pages

Since: 2.10

---

gtk_print_operation_get_n_pages_to_print ()

```
gint
gtk_print_operation_get_n_pages_to_print
                            (GtkPrintOperation *op);
```

Returns the number of pages that will be printed.

Note that this value is set during print preparation phase (`GTK_PRINT_STATUS_PREPARING`), so this function should never be called before the data generation phase (`GTK_PRINT_STATUS_GENERATING_DATA`). You can connect to the “status-changed” signal and call `gtk_print_operation_get_n_pages_to_print()` when print status is `GTK_PRINT_STATUS_GENERATING_DATA`. This is typically used to track the progress of print operation.

Parameters

op	a GtkPrintOperation

Returns

the number of pages that will be printed

Since: 2.18

gtk_print_operation_set_current_page ()

```
void
gtk_print_operation_set_current_page (GtkPrintOperation *op,
                                      gint current_page);
```

Sets the current page.

If this is called before `gtk_print_operation_run()`, the user will be able to select to print only the current page.

Note that this only makes sense for pre-paginated documents.

Parameters

op	a GtkPrintOperation
current_page	the current page, 0-based

Since: 2.10

---

gtk_print_operation_set_use_full_page ()

```
void
gtk_print_operation_set_use_full_page (GtkPrintOperation *op,
                                       gboolean full_page);
```

If `full_page` is `TRUE`, the transformation for the cairo context obtained from GtkPrintContext puts the origin at the top left corner of the page (which may not be the top left corner of the sheet, depending on page orientation and the number of pages per sheet). Otherwise, the origin is at the top left corner of the imageable area (i.e. inside the margins).

Parameters

op	a GtkPrintOperation
full_page	`TRUE` to set up the GtkPrintContext for the full page

Since: 2.10

---

gtk_print_operation_set_unit ()

```
void
gtk_print_operation_set_unit (GtkPrintOperation *op,
                              GtkUnit unit);
```

Sets up the transformation for the cairo context obtained from GtkPrintContext in such a way that distances are measured in units of `unit` .

Parameters

op	a GtkPrintOperation
unit	the unit to use

Since: 2.10

---

gtk_print_operation_set_export_filename ()

```
void
gtk_print_operation_set_export_filename
                                (GtkPrintOperation *op,
                                 const gchar *filename);
```

Sets up the GtkPrintOperation to generate a file instead of showing the print dialog. The indended use of this function is for implementing “Export to PDF” actions. Currently, PDF is the only supported format.

“Print to PDF” support is independent of this and is done by letting the user pick the “Print to PDF” item from the list of printers in the print dialog.

Parameters

op	a GtkPrintOperation	
filename	the filename for the exported file.	[type filename]

Since: 2.10

---

gtk_print_operation_set_show_progress ()

```
void
gtk_print_operation_set_show_progress (GtkPrintOperation *op,
                                       gboolean show_progress);
```

If `show_progress` is `TRUE`, the print operation will show a progress dialog during the print operation.

Parameters

op	a GtkPrintOperation
show_progress	`TRUE` to show a progress dialog

Since: 2.10

---

gtk_print_operation_set_track_print_status ()

```
void
gtk_print_operation_set_track_print_status
                                (GtkPrintOperation *op,
                                 gboolean track_status);
```

If track_status is `TRUE`, the print operation will try to continue report on the status of the print job in the printer queues and printer. This can allow your application to show things like “out of paper” issues, and when the print job actually reaches the printer.

This function is often implemented using some form of polling, so it should not be enabled unless needed.

Parameters

op	a GtkPrintOperation	
track_status	TRUE to track status after printing	

Since: 2.10

---

gtk_print_operation_set_custom_tab_label ()

```
void
gtk_print_operation_set_custom_tab_label
                                (GtkPrintOperation *op,
                                 const gchar *label);
```

Sets the label for the tab holding custom widgets.

Parameters

op	a GtkPrintOperation	
label	the label to use, or NULL to use the default label.	[allow-none]

Since: 2.10

---

gtk_print_operation_run ()

```
GtkPrintOperationResult
gtk_print_operation_run (GtkPrintOperation *op,
                         GtkPrintOperationAction action,
                         GtkWindow *parent,
                         GError **error);
```

Runs the print operation, by first letting the user modify print settings in the print dialog, and then print the document.

Normally that this function does not return until the rendering of all pages is complete. You can connect to the "status-changed" signal on op to obtain some information about the progress of the print operation. Furthermore, it may use a recursive mainloop to show the print dialog.

If you call gtk_print_operation_set_allow_async() or set the "allow-async" property the operation will run asynchronously if this is supported on the platform. The "done" signal will be emitted with the result of the operation when the it is done (i.e. when the dialog is canceled, or when the print succeeds or fails).

```
1                                       if (settings != NULL)
2                                         gtk_print_operation_set_print_settings
3                                       (print, settings);
4
5                                       if (page_setup != NULL)
6                                         gtk_print_operation_set_default_page_setup
7                                       (print, page_setup);
8
9                                       g_signal_connect (print, "begin-print",
```

```
                                                  G_CALLBACK (begin_print),
&data);
g_signal_connect (print, "draw-page",
                                                  G_CALLBACK (draw_page),
&data);

res = gtk_print_operation_run (print,

GTK_PRINT_OPERATION_ACTION_PRINT_DIALOG,
                                                    parent,
                                                    &error);

if (res == GTK_PRINT_OPERATION_RESULT_ERROR)
 {
   error_dialog = gtk_message_dialog_new
(GTK_WINDOW (parent),

GTK_DIALOG_DESTROY_WITH_PARENT,

GTK_MESSAGE_ERROR,

GTK_BUTTONS_CLOSE,
                                                  "Error
printing file:\n%s",
                                                  error-
>message);
   g_signal_connect (error_dialog,
"response",
                         G_CALLBACK
(gtk_widget_destroy), NULL);
   gtk_widget_show (error_dialog);
   g_error_free (error);
 }
else if (res ==
GTK_PRINT_OPERATION_RESULT_APPLY)
 {
   if (settings != NULL)
g_object_unref (settings);
   settings = g_object_ref
(gtk_print_operation_get_print_settings
(print));
 }
```

Note that `gtk_print_operation_run()` can only be called once on a given GtkPrintOperation.

Parameters

op	a GtkPrintOperation	
action	the action to start	
parent	Transient parent of the dialog.	[allow-none]
error	Return location for errors, or `NULL`.	[allow-none]

Returns

the result of the print operation. A return value of `GTK_PRINT_OPERATION_RESULT_APPLY` indicates that the printing was completed successfully. In this case, it is a good idea to obtain the used print settings with `gtk_print_operation_get_print_settings()` and store them for reuse with the next print operation. A value of `GTK_PRINT_OPERATION_RESULT_IN_PROGRESS` means the operation is running asynchronously, and will emit the “done” signal when done.

Since: 2.10

---

gtk_print_operation_cancel ()

```
void
gtk_print_operation_cancel (GtkPrintOperation *op);
```

Cancels a running print operation. This function may be called from a “begin-print”, “paginate” or “draw-page” signal handler to stop the currently running print operation.

Parameters

op	a GtkPrintOperation

Since: 2.10

---

gtk_print_operation_draw_page_finish ()

```
void
gtk_print_operation_draw_page_finish (GtkPrintOperation *op);
```

Signalize that drawing of particular page is complete.

It is called after completion of page drawing (e.g. drawing in another thread). If `gtk_print_operation_set_defer_drawing()` was called before, then this function has to be called by application. In another case it is called by the library itself.

Parameters

op	a GtkPrintOperation

Since: 2.16

---

gtk_print_operation_set_defer_drawing ()

```
void
gtk_print_operation_set_defer_drawing (GtkPrintOperation *op);
```

Sets up the GtkPrintOperation to wait for calling of `gtk_print_operation_draw_page_finish()` from application. It can be used for drawing page in another thread.

This function must be called in the callback of “draw-page” signal.

Parameters

op	a GtkPrintOperation

Since: 2.16

---

gtk_print_operation_get_status ()

```
GtkPrintStatus
gtk_print_operation_get_status (GtkPrintOperation *op);
```

Returns the status of the print operation. Also see gtk_print_operation_get_status_string().

Parameters

op	a GtkPrintOperation

Returns

the status of the print operation

Since: 2.10

---

gtk_print_operation_get_status_string ()

```
const gchar *
gtk_print_operation_get_status_string (GtkPrintOperation *op);
```

Returns a string representation of the status of the print operation. The string is translated and suitable for displaying the print status e.g. in a GtkStatusbar.

Use gtk_print_operation_get_status() to obtain a status value that is suitable for programmatic use.

Parameters

op	a GtkPrintOperation

Returns

a string representation of the status of the print operation

Since: 2.10

---

gtk_print_operation_is_finished ()

```
gboolean
gtk_print_operation_is_finished (GtkPrintOperation *op);
```

A convenience function to find out if the print operation is finished, either successfully (GTK_PRINT_STATUS_FINISHED) or unsuccessfully (GTK_PRINT_STATUS_FINISHED_ABORTED).

Note: when you enable print status tracking the print operation can be in a non-finished state even after done has been called, as the operation status then tracks the print job status on the printer.

Parameters

op	a GtkPrintOperation

Returns

TRUE, if the print operation is finished.

Since: 2.10

---

gtk_print_operation_set_support_selection ()

```
void
gtk_print_operation_set_support_selection
                                (GtkPrintOperation *op,
                                 gboolean support_selection);
```

Sets whether selection is supported by GtkPrintOperation.

Parameters

op	a GtkPrintOperation
support_selection	TRUE to support selection

Since: 2.18

---

gtk_print_operation_get_support_selection ()

```
gboolean
gtk_print_operation_get_support_selection
                                (GtkPrintOperation *op);
```

Gets the value of “support-selection” property.

Parameters

op	a GtkPrintOperation

Returns

whether the application supports print of selection

Since: 2.18

---

gtk_print_operation_set_has_selection ()

```
void
gtk_print_operation_set_has_selection (GtkPrintOperation *op,
                                       gboolean has_selection);
```

Sets whether there is a selection to print.

Application has to set number of pages to which the selection will draw by `gtk_print_operation_set_n_pages()` in a callback of "begin-print".

Parameters

op	a GtkPrintOperation
has_selection	`TRUE` indicates that a selection exists

Since: 2.18

---

gtk_print_operation_get_has_selection ()

```
gboolean
gtk_print_operation_get_has_selection (GtkPrintOperation *op);
```

Gets the value of "has-selection" property.

Parameters

op	a GtkPrintOperation

Returns

whether there is a selection

Since: 2.18

---

gtk_print_operation_set_embed_page_setup ()

```
void
gtk_print_operation_set_embed_page_setup
                                (GtkPrintOperation *op,
                                 gboolean embed);
```

Embed page size combo box and orientation combo box into page setup page. Selected page setup is stored as default page setup in GtkPrintOperation.

Parameters

op	a GtkPrintOperation
embed	`TRUE` to embed page setup selection in the GtkPrintUnixDialog

Since: 2.18

---

gtk_print_operation_get_embed_page_setup ()

```
gboolean
gtk_print_operation_get_embed_page_setup
                                (GtkPrintOperation *op);
```

Gets the value of “embed-page-setup” property.

Parameters

op	a GtkPrintOperation

Returns

whether page setup selection combos are embedded

Since: 2.18

---

gtk_print_run_page_setup_dialog ()

```
GtkPageSetup *
gtk_print_run_page_setup_dialog (GtkWindow *parent,
                                 GtkPageSetup *page_setup,
                                 GtkPrintSettings *settings);
```

Runs a page setup dialog, letting the user modify the values from `page_setup` . If the user cancels the dialog, the returned GtkPageSetup is identical to the passed in `page_setup` , otherwise it contains the modifications done in the dialog.

Note that this function may use a recursive mainloop to show the page setup dialog. See `gtk_print_run_page_setup_dialog_async()` if this is a problem.

Parameters

parent	transient parent.	[allow-none]
page_setup	an existing GtkPageSetup.	[allow-none]
settings	a GtkPrintSettings	

Returns

a new GtkPageSetup.

[transfer full]

Since: 2.10

GtkPageSetupDoneFunc ()

```
void
(*GtkPageSetupDoneFunc) (GtkPageSetup *page_setup,
                         gpointer data);
```

The type of function that is passed to `gtk_print_run_page_setup_dialog_async()`.

This function will be called when the page setup dialog is dismissed, and also serves as destroy notify for `data` .

Parameters

page_setup	the GtkPageSetup that has been	
data	user data that has been passed to `gtk_print_run_page_setup_dialog_async()`.	[closure]

---

gtk_print_run_page_setup_dialog_async ()

```
void
gtk_print_run_page_setup_dialog_async (GtkWindow *parent,
                                       GtkPageSetup *page_setup,
                                       GtkPrintSettings *settings,
                                       GtkPageSetupDoneFunc done_cb,
                                       gpointer data);
```

Runs a page setup dialog, letting the user modify the values from `page_setup` .

In contrast to `gtk_print_run_page_setup_dialog()`, this function returns after showing the page setup dialog on platforms that support this, and calls `done_cb` from a signal handler for the ::response signal of the dialog.

Parameters

parent	transient parent, or `NULL`.	[allow-none]
page_setup	an existing GtkPageSetup, or `NULL`.	[allow-none]
settings	a GtkPrintSettings	
done_cb	a function to call when the user saves the modified page setup.	[scope async]
data	user data to pass to `done_cb`	

Since: 2.10

---

gtk_print_operation_preview_end_preview ()

```
void
gtk_print_operation_preview_end_preview
                                (GtkPrintOperationPreview *preview);
```

Ends a preview.

This function must be called to finish a custom print preview.

Parameters

preview	a GtkPrintOperationPreview

Since: 2.10

---

gtk_print_operation_preview_is_selected ()

```
gboolean
gtk_print_operation_preview_is_selected
                                (GtkPrintOperationPreview *preview,
                                 gint page_nr);
```

Returns whether the given page is included in the set of pages that have been selected for printing.

Parameters

preview	a GtkPrintOperationPreview
page_nr	a page number

Returns

`TRUE` if the page has been selected for printing

Since: 2.10

---

gtk_print_operation_preview_render_page ()

```
void
gtk_print_operation_preview_render_page
                                (GtkPrintOperationPreview *preview,
                                 gint page_nr);
```

Renders a page to the preview, using the print context that was passed to the "preview" handler together with `preview`.

A custom iprint preview should use this function in its ::expose handler to render the currently selected page.

Note that this function requires a suitable cairo context to be associated with the print context.

Parameters

preview	a GtkPrintOperationPreview
page_nr	the page to render

Since: 2.10

Types and Values

struct GtkPrintOperation

```
struct GtkPrintOperation;
```

---

struct GtkPrintOperationClass

```
struct GtkPrintOperationClass {
  GObjectClass parent_class;


  void     (*done)               (GtkPrintOperation *operation,
                                  GtkPrintOperationResult result);
  void     (*begin_print)        (GtkPrintOperation *operation,
                                  GtkPrintContext   *context);
  gboolean (*paginate)           (GtkPrintOperation *operation,
                                  GtkPrintContext   *context);
  void     (*request_page_setup) (GtkPrintOperation *operation,
                                  GtkPrintContext   *context,
                                  gint               page_nr,
                                  GtkPageSetup      *setup);
  void     (*draw_page)          (GtkPrintOperation *operation,
                                  GtkPrintContext   *context,
                                  gint               page_nr);
  void     (*end_print)          (GtkPrintOperation *operation,
                                  GtkPrintContext   *context);
  void     (*status_changed)     (GtkPrintOperation *operation);

  GtkWidget *(*create_custom_widget) (GtkPrintOperation *operation);
  void       (*custom_widget_apply)  (GtkPrintOperation *operation,
                                      GtkWidget         *widget);

  gboolean (*preview)        (GtkPrintOperation        *operation,
                              GtkPrintOperationPreview *preview,
                              GtkPrintContext          *context,
                              GtkWindow                *parent);

  void     (*update_custom_widget) (GtkPrintOperation *operation,
                                    GtkWidget         *widget,
                                    GtkPageSetup      *setup,
                                    GtkPrintSettings  *settings);
};
```

Members

`done` ()	Signal emitted when the print operation run has finished doing everything required for printing.
`begin_print` ()	Signal emitted after the user has finished changing print settings in the dialog, before the actual rendering starts.
`paginate` ()	Signal emitted after the “begin-print” signal, but before the actual rendering starts.
`request_page_setup` ()	Emitted once for every page that is printed, to give the application a chance to modify the page setup.
`draw_page` ()	Signal emitted for every page that is printed.
`end_print` ()	Signal emitted after all pages have

	been rendered.
`status_changed` ()	Emitted at between the various phases of the print operation.
`create_custom_widget` ()	Signal emitted when displaying the print dialog.
`custom_widget_apply` ()	Signal emitted right before “begin-print” if you added a custom widget in the “create-custom-widget” handler.
`preview` ()	Signal emitted when a preview is requested from the native dialog.
`update_custom_widget` ()	Emitted after change of selected printer.

---

enum GtkPrintStatus

The status gives a rough indication of the completion of a running print operation.

Members

GTK_PRINT_STATUS_INITIAL	The printing has not started yet; this status is set initially, and while the print dialog is shown.
GTK_PRINT_STATUS_PREPARING	This status is set while the begin-print signal is emitted and during pagination.
GTK_PRINT_STATUS_GENERATING_DATA	This status is set while the pages are being rendered.
GTK_PRINT_STATUS_SENDING_DATA	The print job is being sent off to the printer.
GTK_PRINT_STATUS_PENDING	The print job has been sent to the printer, but is not printed for some reason, e.g. the printer may be stopped.
GTK_PRINT_STATUS_PENDING_ISSUE	Some problem has occurred during printing, e.g. a paper jam.
GTK_PRINT_STATUS_PRINTING	The printer is processing the print job.
GTK_PRINT_STATUS_FINISHED	The printing has been completed successfully.
GTK_PRINT_STATUS_FINISHED_ABORTED	The printing has been aborted.

---

enum GtkPrintOperationAction

The `action` parameter to `gtk_print_operation_run()` determines what action the print operation should perform.

Members

GTK_PRINT_OPERATION_ACTION_PRINT_DIALOG	Show the print dialog.
GTK_PRINT_OPERATION_ACTION_PRINT	Start to print without showing the print dialog, based on the current print settings.
GTK_PRINT_OPERATION_ACTION_PREVIEW	Show the print preview.
GTK_PRINT_OPERATION_ACTION_EXPORT	Export to a file. This requires the export-filename property to be set.

---

enum GtkPrintOperationResult

A value of this type is returned by `gtk_print_operation_run()`.

Members

GTK_PRINT_OPERATION_RESULT_ERROR	An error has occurred.
GTK_PRINT_OPERATION_RESULT_APPLY	The print settings should be stored.
GTK_PRINT_OPERATION_RESULT_CANCEL	The print operation has been canceled, the print settings should not be stored.
GTK_PRINT_OPERATION_RESULT_IN_PROGRESS	The print operation is not complete yet. This value will only be returned when running asynchronously.

---

enum GtkPrintError

Error codes that identify various errors that can occur while using the GTK+ printing support.

Members

GTK_PRINT_ERROR_GENERAL	An unspecified error occurred.
GTK_PRINT_ERROR_INTERNAL_ERROR	An internal error occurred.
GTK_PRINT_ERROR_NOMEM	A memory allocation failed.
GTK_PRINT_ERROR_INVALID_FILE	An error occurred while loading a page setup or paper size from a key file.

GTK_PRINT_ERROR

```
#define GTK_PRINT_ERROR gtk_print_error_quark ()
```

The error domain for GtkPrintError errors.

GtkPrintOperationPreview

```
typedef struct _GtkPrintOperationPreview GtkPrintOperationPreview;
```

Property Details

The “allow-async” property

```
  “allow-async”              gboolean
```

Determines whether the print operation may run asynchronously or not.

Some systems don't support asynchronous printing, but those that do will return GTK_PRINT_OPERATION_RESULT_IN_PROGRESS as the status, and emit the “done” signal when the operation is actually done.

The Windows port does not support asynchronous operation at all (this is unlikely to change). On other platforms, all actions except for GTK_PRINT_OPERATION_ACTION_EXPORT support asynchronous operation.

Flags: Read / Write

Default value: FALSE

Since: 2.10

The “current-page” property

```
  “current-page”             gint
```

The current page in the document.

If this is set before gtk_print_operation_run(), the user will be able to select to print only the current page.

Note that this only makes sense for pre-paginated documents.

Flags: Read / Write

Allowed values: >= -1

Default value: -1

Since: 2.10

The “custom-tab-label” property

“custom-tab-label” gchar *

Used as the label of the tab containing custom widgets. Note that this property may be ignored on some platforms.

If this is NULL, GTK+ uses a default label.

Flags: Read / Write

Default value: NULL

Since: 2.10

The “default-page-setup” property

“default-page-setup” GtkPageSetup *

The GtkPageSetup used by default.

This page setup will be used by gtk_print_operation_run(), but it can be overridden on a per-page basis by connecting to the “request-page-setup” signal.

Flags: Read / Write

Since: 2.10

The “embed-page-setup” property

“embed-page-setup” gboolean

If TRUE, page size combo box and orientation combo box are embedded into page setup page.

Flags: Read / Write

Default value: FALSE

Since: 2.18

The “export-filename” property

“export-filename” gchar *

The name of a file to generate instead of showing the print dialog. Currently, PDF is the only supported format.

The intended use of this property is for implementing “Export to PDF” actions.

“Print to PDF” support is independent of this and is done by letting the user pick the “Print to PDF” item from the list of printers in the print dialog.

Flags: Read / Write

Default value: NULL

Since: 2.10

The “has-selection” property

“has-selection” gboolean

Determines whether there is a selection in your application. This can allow your application to print the selection. This is typically used to make a "Selection" button sensitive.

Flags: Read / Write

Default value: FALSE

Since: 2.18

The “job-name” property

“job-name” gchar *

A string used to identify the job (e.g. in monitoring applications like eggcups).

If you don't set a job name, GTK+ picks a default one by numbering successive print jobs.

Flags: Read / Write

Default value: ""

Since: 2.10

The “n-pages” property

“n-pages” gint

The number of pages in the document.

This must be set to a positive number before the rendering starts. It may be set in a “begin-print” signal hander.

Note that the page numbers passed to the “request-page-setup” and “draw-page” signals are 0-based, i.e. if the user chooses to print all pages, the last ::draw-page signal will be for page `n_pages` - 1.

Flags: Read / Write

Allowed values: >= -1

Default value: -1

Since: 2.10

The “n-pages-to-print” property

“n-pages-to-print” gint

The number of pages that will be printed.

Note that this value is set during print preparation phase (`GTK_PRINT_STATUS_PREPARING`), so this value should never be get before the data generation phase (`GTK_PRINT_STATUS_GENERATING_DATA`). You can connect to the

“status-changed” signal and call gtk_print_operation_get_n_pages_to_print() when print status is GTK_PRINT_STATUS_GENERATING_DATA. This is typically used to track the progress of print operation.

Flags: Read

Allowed values: >= -1

Default value: -1

Since: 2.18

The “print-settings” property

“print-settings” GtkPrintSettings *

The GtkPrintSettings used for initializing the dialog.

Setting this property is typically used to re-establish print settings from a previous print operation, see gtk_print_operation_run().

Flags: Read / Write

Since: 2.10

The “show-progress” property

“show-progress” gboolean

Determines whether to show a progress dialog during the print operation.

Flags: Read / Write

Default value: FALSE

Since: 2.10

The “status” property

“status” GtkPrintStatus

The status of the print operation.

Flags: Read

Default value: GTK_PRINT_STATUS_INITIAL

Since: 2.10

The “status-string” property

“status-string” gchar *

A string representation of the status of the print operation. The string is translated and suitable for displaying the

print status e.g. in a GtkStatusbar.

See the “status” property for a status value that is suitable for programmatic use.

Flags: Read

Default value: ""

Since: 2.10

The “support-selection” property

“support-selection” gboolean

If TRUE, the print operation will support print of selection. This allows the print dialog to show a "Selection" button.

Flags: Read / Write

Default value: FALSE

Since: 2.18

The “track-print-status” property

“track-print-status” gboolean

If TRUE, the print operation will try to continue report on the status of the print job in the printer queues and printer. This can allow your application to show things like “out of paper” issues, and when the print job actually reaches the printer. However, this is often implemented using polling, and should not be enabled unless needed.

Flags: Read / Write

Default value: FALSE

Since: 2.10

The “unit” property

“unit” GtkUnit

The transformation for the cairo context obtained from GtkPrintContext is set up in such a way that distances are measured in units of unit .

Flags: Read / Write

Default value: GTK_UNIT_NONE

Since: 2.10

The “use-full-page” property

```
  “use-full-page”            gboolean
```

If TRUE, the transformation for the cairo context obtained from GtkPrintContext puts the origin at the top left corner of the page (which may not be the top left corner of the sheet, depending on page orientation and the number of pages per sheet). Otherwise, the origin is at the top left corner of the imageable area (i.e. inside the margins).

Flags: Read / Write

Default value: FALSE

Since: 2.10

Signal Details

The “begin-print” signal

```
void
user_function (GtkPrintOperation *operation,
               GtkPrintContext   *context,
               gpointer           user_data)
```

Emitted after the user has finished changing print settings in the dialog, before the actual rendering starts.

A typical use for ::begin-print is to use the parameters from the GtkPrintContext and paginate the document accordingly, and then set the number of pages with `gtk_print_operation_set_n_pages()`.

Parameters

operation	the GtkPrintOperation on which the signal was emitted
context	the GtkPrintContext for the current operation
user_data	user data set when the signal handler was connected.

Flags: Run Last

Since: 2.10

The “create-custom-widget” signal

```
GObject*
user_function (GtkPrintOperation *operation,
               gpointer           user_data)
```

Emitted when displaying the print dialog. If you return a widget in a handler for this signal it will be added to a custom tab in the print dialog. You typically return a container widget with multiple widgets in it.

The print dialog owns the returned widget, and its lifetime is not controlled by the application. However, the widget is guaranteed to stay around until the “custom-widget-apply” signal is emitted on the operation. Then you can read out any information you need from the widgets.

Parameters

operation	the GtkPrintOperation on which the signal was emitted
user_data	user data set when the signal handler was connected.

Returns

A custom widget that gets embedded in the print dialog, or `NULL`.

[transfer none]

Flags: Run Last

Since: 2.10

The `"custom-widget-apply"` signal

```
void
user_function (GtkPrintOperation *operation,
               GtkWidget         *widget,
               gpointer           user_data)
```

Emitted right before "begin-print" if you added a custom widget in the "create-custom-widget" handler. When you get this signal you should read the information from the custom widgets, as the widgets are not guaraneed to be around at a later time.

Parameters

operation	the GtkPrintOperation on which the signal was emitted
widget	the custom widget added in create-custom-widget
user_data	user data set when the signal handler was connected.

Flags: Run Last

Since: 2.10

The `"done"` signal

```
void
user_function (GtkPrintOperation      *operation,
               GtkPrintOperationResult result,
               gpointer                user_data)
```

Emitted when the print operation run has finished doing everything required for printing.

`result` gives you information about what happened during the run. If `result` is

GTK_PRINT_OPERATION_RESULT_ERROR then you can call gtk_print_operation_get_error() for more information.

If you enabled print status tracking then gtk_print_operation_is_finished() may still return FALSE after "done" was emitted.

Parameters

operation	the GtkPrintOperation on which the signal was emitted
result	the result of the print operation
user_data	user data set when the signal handler was connected.

Flags: Run Last

Since: 2.10

The "draw-page" signal

```
void
user_function (GtkPrintOperation *operation,
               GtkPrintContext   *context,
               gint               page_nr,
               gpointer           user_data)
```

Emitted for every page that is printed. The signal handler must render the page_nr 's page onto the cairo context obtained from context using gtk_print_context_get_cairo_context().

```
1                                               static void
2                                               draw_page (GtkPrintOperation *operation,
3                                                          GtkPrintContext   *context,
4                                                          gint               page_nr,
5                                                          gpointer           user_data)
6                                               {
7                                                 cairo_t *cr;
8                                                 PangoLayout *layout;
9                                                 gdouble width, text_height;
10                                                gint layout_height;
11                                                PangoFontDescription *desc;
12
13                                                cr = gtk_print_context_get_cairo_context
14                                              (context);
15                                                width = gtk_print_context_get_width
16                                              (context);
17
18                                                cairo_rectangle (cr, 0, 0, width,
19                                              HEADER_HEIGHT);
20
21                                                cairo_set_source_rgb (cr, 0.8, 0.8, 0.8);
22                                                cairo_fill (cr);
23
24                                                layout =
25                                              gtk_print_context_create_pango_layout
26                                              (context);
27
28                                                desc = pango_font_description_from_string
29                                              ("sans 14");
```

```
30                                                  pango_layout_set_font_description (layout,
31                                                desc);
32                                                  pango_font_description_free (desc);
33
34                                                  pango_layout_set_text (layout, "some text",
35                                                -1);
36                                                  pango_layout_set_width (layout, width *
37                                                PANGO_SCALE);
38                                                  pango_layout_set_alignment (layout,
                                                  PANGO_ALIGN_CENTER);

                                                    pango_layout_get_size (layout, NULL,
                                                  &layout_height);
                                                    text_height = (gdouble)layout_height /
                                                  PANGO_SCALE;

                                                    cairo_move_to (cr, width / 2,
                                                  (HEADER_HEIGHT - text_height) / 2);
                                                    pango_cairo_show_layout (cr, layout);

                                                    g_object_unref (layout);
                                                  }
```

Use `gtk_print_operation_set_use_full_page()` and `gtk_print_operation_set_unit()` before starting the print operation to set up the transformation of the cairo context according to your needs.

Parameters

operation	the GtkPrintOperation on which the signal was emitted
context	the GtkPrintContext for the current operation
page_nr	the number of the currently printed page (0-based)
user_data	user data set when the signal handler was connected.

Flags: Run Last

Since: 2.10

The `"end-print"` signal

```
void
user_function (GtkPrintOperation *operation,
               GtkPrintContext   *context,
               gpointer           user_data)
```

Emitted after all pages have been rendered. A handler for this signal can clean up any resources that have been allocated in the "begin-print" handler.

Parameters

operation	the GtkPrintOperation on which the signal was emitted
context	the GtkPrintContext for the current

	operation
user_data	user data set when the signal handler was connected.

Flags: Run Last

Since: 2.10

The “paginate” signal

```
gboolean
user_function (GtkPrintOperation *operation,
               GtkPrintContext   *context,
               gpointer           user_data)
```

Emitted after the “begin-print” signal, but before the actual rendering starts. It keeps getting emitted until a connected signal handler returns TRUE.

The ::paginate signal is intended to be used for paginating a document in small chunks, to avoid blocking the user interface for a long time. The signal handler should update the number of pages using gtk_print_operation_set_n_pages(), and return TRUE if the document has been completely paginated.

If you don't need to do pagination in chunks, you can simply do it all in the ::begin-print handler, and set the number of pages from there.

Parameters

operation	the GtkPrintOperation on which the signal was emitted
context	the GtkPrintContext for the current operation
user_data	user data set when the signal handler was connected.

Returns

TRUE if pagination is complete

Flags: Run Last

Since: 2.10

The “preview” signal

```
gboolean
user_function (GtkPrintOperation        *operation,
               GtkPrintOperationPreview *preview,
               GtkPrintContext          *context,
               GtkWindow                *parent,
               gpointer                  user_data)
```

Gets emitted when a preview is requested from the native dialog.

The default handler for this signal uses an external viewer application to preview.

To implement a custom print preview, an application must return `TRUE` from its handler for this signal. In order to use the provided `context` for the preview implementation, it must be given a suitable cairo context with `gtk_print_context_set_cairo_context()`.

The custom preview implementation can use `gtk_print_operation_preview_is_selected()` and `gtk_print_operation_preview_render_page()` to find pages which are selected for print and render them. The preview must be finished by calling `gtk_print_operation_preview_end_preview()` (typically in response to the user clicking a close button).

Parameters

operation	the GtkPrintOperation on which the signal was emitted	
preview	the GtkPrintOperationPreview for the current operation	
context	the GtkPrintContext that will be used	
parent	the GtkWindow to use as window parent, or `NULL`.	[allow-none]
user_data	user data set when the signal handler was connected.	

Returns

`TRUE` if the listener wants to take over control of the preview

Flags: Run Last

Since: 2.10

The `"request-page-setup"` signal

```
void
user_function (GtkPrintOperation *operation,
               GtkPrintContext   *context,
               gint               page_nr,
               GtkPageSetup      *setup,
               gpointer           user_data)
```

Emitted once for every page that is printed, to give the application a chance to modify the page setup. Any changes done to `setup` will be in force only for printing this page.

Parameters

operation	the GtkPrintOperation on which the signal was emitted	
context	the GtkPrintContext for the current operation	
page_nr	the number of the currently printed page (0-based)	

setup	the GtkPageSetup
user_data	user data set when the signal handler was connected.

Flags: Run Last

Since: 2.10

The `"status-changed"` signal

```
void
user_function (GtkPrintOperation *operation,
               gpointer           user_data)
```

Emitted at between the various phases of the print operation. See GtkPrintStatus for the phases that are being discriminated. Use `gtk_print_operation_get_status()` to find out the current status.

Parameters

operation	the GtkPrintOperation on which the signal was emitted
user_data	user data set when the signal handler was connected.

Flags: Run Last

Since: 2.10

The `"update-custom-widget"` signal

```
void
user_function (GtkPrintOperation *operation,
               GtkWidget         *widget,
               GtkPageSetup      *setup,
               GtkPrintSettings  *settings,
               gpointer           user_data)
```

Emitted after change of selected printer. The actual page setup and print settings are passed to the custom widget, which can actualize itself according to this change.

Parameters

operation	the GtkPrintOperation on which the signal was emitted
widget	the custom widget added in create-custom-widget
setup	actual page setup
settings	actual print settings
user_data	user data set when the signal handler was connected.

Flags: Run Last

Since: 2.18

The “got-page-size” signal

```
void
user_function (GtkPrintOperationPreview *preview,
               GtkPrintContext          *context,
               GtkPageSetup             *page_setup,
               gpointer                  user_data)
```

The ::got-page-size signal is emitted once for each page that gets rendered to the preview.

A handler for this signal should update the `context` according to `page_setup` and set up a suitable cairo context, using `gtk_print_context_set_cairo_context()`.

Parameters

preview	the object on which the signal is emitted
context	the current GtkPrintContext
page_setup	the GtkPageSetup for the current page
user_data	user data set when the signal handler was connected.

Flags: Run Last

The “ready” signal

```
void
user_function (GtkPrintOperationPreview *preview,
               GtkPrintContext          *context,
               gpointer                  user_data)
```

The ::ready signal gets emitted once per preview operation, before the first page is rendered.

A handler for this signal can be used for setup tasks.

Parameters

preview	the object on which the signal is emitted
context	the current GtkPrintContext
user_data	user data set when the signal handler was connected.

Flags: Run Last

See Also

GtkPrintContext, GtkPrintUnixDialog

GtkPrintContext

GtkPrintContext — Encapsulates context for drawing pages

Functions

cairo_t *	gtk_print_context_get_cairo_context ()
void	gtk_print_context_set_cairo_context ()
GtkPageSetup *	gtk_print_context_get_page_setup ()
gdouble	gtk_print_context_get_width ()
gdouble	gtk_print_context_get_height ()
gdouble	gtk_print_context_get_dpi_x ()
gdouble	gtk_print_context_get_dpi_y ()
PangoFontMap *	gtk_print_context_get_pango_fontmap ()
PangoContext *	gtk_print_context_create_pango_context ()
PangoLayout *	gtk_print_context_create_pango_layout ()
gboolean	gtk_print_context_get_hard_margins ()

Types and Values

GtkPrintContext

Object Hierarchy

```
GObject
└── GtkPrintContext
```

Includes

```
#include <gtk/gtk.h>
```

Description

A GtkPrintContext encapsulates context information that is required when drawing pages for printing, such as the cairo context and important parameters like page size and resolution. It also lets you easily create PangoLayout and PangoContext objects that match the font metrics of the cairo surface.

GtkPrintContext objects gets passed to the “begin-print”, “end-print”, “request-page-setup” and “draw-page” signals on the GtkPrintOperation.

Using GtkPrintContext in a “draw-page” callback

```
1
2
3
4
5
6
```

```
static void
draw_page (GtkPrintOperation *operation,
           GtkPrintContext   *context,
           int                page_nr)
{
  cairo_t *cr;
```

```
  PangoLayout *layout;
  PangoFontDescription *desc;

  cr = gtk_print_context_get_cairo_context
(context);

  // Draw a red rectangle, as wide as the
paper (inside the margins)
  cairo_set_source_rgb (cr, 1.0, 0, 0);
  cairo_rectangle (cr, 0, 0,
gtk_print_context_get_width (context), 50);

  cairo_fill (cr);

  // Draw some lines
  cairo_move_to (cr, 20, 10);
  cairo_line_to (cr, 40, 20);
  cairo_arc (cr, 60, 60, 20, 0, M_PI);
  cairo_line_to (cr, 80, 20);

  cairo_set_source_rgb (cr, 0, 0, 0);
  cairo_set_line_width (cr, 5);
  cairo_set_line_cap (cr,
CAIRO_LINE_CAP_ROUND);
  cairo_set_line_join (cr,
CAIRO_LINE_JOIN_ROUND);

  cairo_stroke (cr);

  // Draw some text
  layout =
gtk_print_context_create_pango_layout
(context);
  pango_layout_set_text (layout, "Hello
World! Printing is easy", -1);
  desc = pango_font_description_from_string
("sans 28");
  pango_layout_set_font_description (layout,
desc);
  pango_font_description_free (desc);

  cairo_move_to (cr, 30, 20);
  pango_cairo_layout_path (cr, layout);

  // Font Outline
  cairo_set_source_rgb (cr, 0.93, 1.0, 0.47);
  cairo_set_line_width (cr, 0.5);
  cairo_stroke_preserve (cr);

  // Font Fill
  cairo_set_source_rgb (cr, 0, 0.0, 1.0);
  cairo_fill (cr);

  g_object_unref (layout);
}
```

Printing support was added in GTK+ 2.10.

Functions

gtk_print_context_get_cairo_context ()

```
cairo_t *
gtk_print_context_get_cairo_context (GtkPrintContext *context);
```

Obtains the cairo context that is associated with the GtkPrintContext.

Parameters

context	a GtkPrintContext

Returns

the cairo context of `context` .

[transfer none]

Since: 2.10

---

gtk_print_context_set_cairo_context ()

```
void
gtk_print_context_set_cairo_context (GtkPrintContext *context,
                                     cairo_t *cr,
                                     double dpi_x,
                                     double dpi_y);
```

Sets a new cairo context on a print context.

This function is intended to be used when implementing an internal print preview, it is not needed for printing, since GTK+ itself creates a suitable cairo context in that case.

Parameters

context	a GtkPrintContext
cr	the cairo context
dpi_x	the horizontal resolution to use with `cr`
dpi_y	the vertical resolution to use with `cr`

Since: 2.10

---

gtk_print_context_get_page_setup ()

```
GtkPageSetup *
gtk_print_context_get_page_setup (GtkPrintContext *context);
```

Obtains the GtkPageSetup that determines the page dimensions of the GtkPrintContext.

Parameters

context	a GtkPrintContext

Returns

the page setup of `context` .

[transfer none]

Since: 2.10

---

gtk_print_context_get_width ()

```
gdouble
gtk_print_context_get_width (GtkPrintContext *context);
```

Obtains the width of the GtkPrintContext, in pixels.

Parameters

context	a GtkPrintContext

Returns

the width of `context`

Since: 2.10

---

gtk_print_context_get_height ()

```
gdouble
gtk_print_context_get_height (GtkPrintContext *context);
```

Obtains the height of the GtkPrintContext, in pixels.

Parameters

context	a GtkPrintContext

Returns

the height of `context`

Since: 2.10

---

gtk_print_context_get_dpi_x ()

```
gdouble
gtk_print_context_get_dpi_x (GtkPrintContext *context);
```

Obtains the horizontal resolution of the GtkPrintContext, in dots per inch.

Parameters

context	a GtkPrintContext

Returns

the horizontal resolution of `context`

Since: 2.10

---

gtk_print_context_get_dpi_y ()

```
gdouble
gtk_print_context_get_dpi_y (GtkPrintContext *context);
```

Obtains the vertical resolution of the GtkPrintContext, in dots per inch.

Parameters

context	a GtkPrintContext

Returns

the vertical resolution of `context`

Since: 2.10

---

gtk_print_context_get_pango_fontmap ()

```
PangoFontMap *
gtk_print_context_get_pango_fontmap (GtkPrintContext *context);
```

Returns a PangoFontMap that is suitable for use with the GtkPrintContext.

Parameters

context	a GtkPrintContext

Returns

the font map of `context` .

[transfer none]

Since: 2.10

---

gtk_print_context_create_pango_context ()

```
PangoContext *
gtk_print_context_create_pango_context
                                (GtkPrintContext *context);
```

Creates a new PangoContext that can be used with the GtkPrintContext.

Parameters

context	a GtkPrintContext

Returns

a new Pango context for `context` .

[transfer full]

Since: 2.10

---

gtk_print_context_create_pango_layout ()

```
PangoLayout *
gtk_print_context_create_pango_layout (GtkPrintContext *context);
```

Creates a new PangoLayout that is suitable for use with the GtkPrintContext.

Parameters

context	a GtkPrintContext

Returns

a new Pango layout for `context` .

[transfer full]

Since: 2.10

---

gtk_print_context_get_hard_margins ()

```
gboolean
gtk_print_context_get_hard_margins (GtkPrintContext *context,
                                    gdouble *top,
                                    gdouble *bottom,
                                    gdouble *left,
                                    gdouble *right);
```

Obtains the hardware printer margins of the GtkPrintContext, in units.

Parameters

context	a GtkPrintContext	
top	top hardware printer margin.	[out]
bottom	bottom hardware printer margin.	[out]
left	left hardware printer margin.	[out]
right	right hardware printer margin.	[out]

Returns

TRUE if the hard margins were retrieved

Since: 2.20

Types and Values

GtkPrintContext

```
typedef struct _GtkPrintContext GtkPrintContext;
```

GtkPrintSettings

GtkPrintSettings — Stores print settings

Functions

void	(*GtkPrintSettingsFunc) ()
GtkPrintSettings *	gtk_print_settings_new ()
GtkPrintSettings *	gtk_print_settings_copy ()
gboolean	gtk_print_settings_has_key ()
const gchar *	gtk_print_settings_get ()
void	gtk_print_settings_set ()
void	gtk_print_settings_unset ()
void	gtk_print_settings_foreach ()
gboolean	gtk_print_settings_get_bool ()
void	gtk_print_settings_set_bool ()
gdouble	gtk_print_settings_get_double ()
gdouble	gtk_print_settings_get_double_with_default ()
void	gtk_print_settings_set_double ()
gdouble	gtk_print_settings_get_length ()
void	gtk_print_settings_set_length ()
gint	gtk_print_settings_get_int ()
gint	gtk_print_settings_get_int_with_default ()
void	gtk_print_settings_set_int ()
const gchar *	gtk_print_settings_get_printer ()
void	gtk_print_settings_set_printer ()
GtkPageOrientation	gtk_print_settings_get_orientation ()

void	gtk_print_settings_set_orientation ()
GtkPaperSize *	gtk_print_settings_get_paper_size ()
void	gtk_print_settings_set_paper_size ()
gdouble	gtk_print_settings_get_paper_width ()
void	gtk_print_settings_set_paper_width ()
gdouble	gtk_print_settings_get_paper_height ()
void	gtk_print_settings_set_paper_height ()
gboolean	gtk_print_settings_get_use_color ()
void	gtk_print_settings_set_use_color ()
gboolean	gtk_print_settings_get_collate ()
void	gtk_print_settings_set_collate ()
gboolean	gtk_print_settings_get_reverse ()
void	gtk_print_settings_set_reverse ()
GtkPrintDuplex	gtk_print_settings_get_duplex ()
void	gtk_print_settings_set_duplex ()
GtkPrintQuality	gtk_print_settings_get_quality ()
void	gtk_print_settings_set_quality ()
gint	gtk_print_settings_get_n_copies ()
void	gtk_print_settings_set_n_copies ()
gint	gtk_print_settings_get_number_up ()
void	gtk_print_settings_set_number_up ()
GtkNumberUpLayout	gtk_print_settings_get_number_up_layout ()
void	gtk_print_settings_set_number_up_layout ()
gint	gtk_print_settings_get_resolution ()
void	gtk_print_settings_set_resolution ()
void	gtk_print_settings_set_resolution_xy ()
gint	gtk_print_settings_get_resolution_x ()
gint	gtk_print_settings_get_resolution_y ()
gdouble	gtk_print_settings_get_printer_lpi ()
void	gtk_print_settings_set_printer_lpi ()
gdouble	gtk_print_settings_get_scale ()
void	gtk_print_settings_set_scale ()
GtkPrintPages	gtk_print_settings_get_print_pages ()
void	gtk_print_settings_set_print_pages ()
GtkPageRange *	gtk_print_settings_get_page_ranges ()
void	gtk_print_settings_set_page_ranges ()
GtkPageSet	gtk_print_settings_get_page_set ()
void	gtk_print_settings_set_page_set ()
const gchar *	gtk_print_settings_get_default_source ()
void	gtk_print_settings_set_default_source ()
const gchar *	gtk_print_settings_get_media_type ()
void	gtk_print_settings_set_media_type ()
const gchar *	gtk_print_settings_get_dither ()
void	gtk_print_settings_set_dither ()
const gchar *	gtk_print_settings_get_finishings ()
void	gtk_print_settings_set_finishings ()
const gchar *	gtk_print_settings_get_output_bin ()
void	gtk_print_settings_set_output_bin ()
GtkPrintSettings *	gtk_print_settings_new_from_file ()
GtkPrintSettings *	gtk_print_settings_new_from_key_file ()
GtkPrintSettings *	gtk_print_settings_new_from_gvariant ()
gboolean	gtk_print_settings_load_file ()

gboolean	gtk_print_settings_load_key_file ()
gboolean	gtk_print_settings_to_file ()
void	gtk_print_settings_to_key_file ()
GVariant *	gtk_print_settings_to_gvariant ()

Types and Values

	GtkPrintSettings
#define	GTK_PRINT_SETTINGS_PRINTER
enum	GtkPageOrientation
#define	GTK_PRINT_SETTINGS_ORIENTATION
#define	GTK_PRINT_SETTINGS_PAPER_FORMAT
#define	GTK_PRINT_SETTINGS_PAPER_WIDTH
#define	GTK_PRINT_SETTINGS_PAPER_HEIGHT
#define	GTK_PRINT_SETTINGS_USE_COLOR
#define	GTK_PRINT_SETTINGS_COLLATE
#define	GTK_PRINT_SETTINGS_REVERSE
enum	GtkPrintDuplex
#define	GTK_PRINT_SETTINGS_DUPLEX
enum	GtkPrintQuality
#define	GTK_PRINT_SETTINGS_QUALITY
#define	GTK_PRINT_SETTINGS_N_COPIES
#define	GTK_PRINT_SETTINGS_NUMBER_UP
enum	GtkNumberUpLayout
#define	GTK_PRINT_SETTINGS_NUMBER_UP_LAYOUT
#define	GTK_PRINT_SETTINGS_RESOLUTION
#define	GTK_PRINT_SETTINGS_RESOLUTION_X
#define	GTK_PRINT_SETTINGS_RESOLUTION_Y
#define	GTK_PRINT_SETTINGS_PRINTER_LPI
#define	GTK_PRINT_SETTINGS_SCALE
enum	GtkPrintPages
#define	GTK_PRINT_SETTINGS_PRINT_PAGES
struct	GtkPageRange
#define	GTK_PRINT_SETTINGS_PAGE_RANGES
enum	GtkPageSet
#define	GTK_PRINT_SETTINGS_PAGE_SET
#define	GTK_PRINT_SETTINGS_DEFAULT_SOURCE
#define	GTK_PRINT_SETTINGS_MEDIA_TYPE
#define	GTK_PRINT_SETTINGS_DITHER
#define	GTK_PRINT_SETTINGS_FINISHINGS
#define	GTK_PRINT_SETTINGS_OUTPUT_BIN
#define	GTK_PRINT_SETTINGS_OUTPUT_DIR
#define	GTK_PRINT_SETTINGS_OUTPUT_BASENAME
#define	GTK_PRINT_SETTINGS_OUTPUT_FILE_FORMAT
#define	GTK_PRINT_SETTINGS_OUTPUT_URI
#define	GTK_PRINT_SETTINGS_WIN32_DRIVER_EXTRA
#define	GTK_PRINT_SETTINGS_WIN32_DRIVER_VERSION

Object Hierarchy

```
GObject
└── GtkPrintSettings
```

Includes

```
#include <gtk/gtk.h>
```

Description

A GtkPrintSettings object represents the settings of a print dialog in a system-independent way. The main use for this object is that once you've printed you can get a settings object that represents the settings the user chose, and the next time you print you can pass that object in so that the user doesn't have to re-set all his settings.

Its also possible to enumerate the settings so that you can easily save the settings for the next time your app runs, or even store them in a document. The predefined keys try to use shared values as much as possible so that moving such a document between systems still works.

Printing support was added in GTK+ 2.10.

Functions

GtkPrintSettingsFunc ()

```
void
(*GtkPrintSettingsFunc) (const gchar *key,
                         const gchar *value,
                         gpointer user_data);
```

---

gtk_print_settings_new ()

```
GtkPrintSettings *
gtk_print_settings_new (void);
```

Creates a new GtkPrintSettings object.

Returns

a new GtkPrintSettings object

Since: 2.10

---

gtk_print_settings_copy ()

```
GtkPrintSettings *
gtk_print_settings_copy (GtkPrintSettings *other);
```

Copies a GtkPrintSettings object.

Parameters

other	a GtkPrintSettings

Returns

a newly allocated copy of `other` .

[transfer full]

Since: 2.10

---

gtk_print_settings_has_key ()

```
gboolean
gtk_print_settings_has_key (GtkPrintSettings *settings,
                            const gchar *key);
```

Returns `TRUE`, if a value is associated with `key` .

Parameters

settings	a GtkPrintSettings
key	a key

Returns

`TRUE`, if `key` has a value

Since: 2.10

---

gtk_print_settings_get ()

```
const gchar *
gtk_print_settings_get (GtkPrintSettings *settings,
                        const gchar *key);
```

Looks up the string value associated with `key` .

Parameters

settings	a GtkPrintSettings
key	a key

Returns

the string value for `key`

Since: 2.10

---

gtk_print_settings_set ()

```
void
gtk_print_settings_set (GtkPrintSettings *settings,
                        const gchar *key,
                        const gchar *value);
```

Associates `value` with `key` .

Parameters

settings	a GtkPrintSettings	
key	a key	
value	a string value, or `NULL`.	[allow-none]

Since: 2.10

---

gtk_print_settings_unset ()

```
void
gtk_print_settings_unset (GtkPrintSettings *settings,
                          const gchar *key);
```

Removes any value associated with `key` . This has the same effect as setting the value to `NULL`.

Parameters

settings	a GtkPrintSettings
key	a key

Since: 2.10

---

gtk_print_settings_foreach ()

```
void
gtk_print_settings_foreach (GtkPrintSettings *settings,
                            GtkPrintSettingsFunc func,
                            gpointer user_data);
```

Calls `func` for each key-value pair of `settings` .

Parameters

settings	a GtkPrintSettings

func	the function to call.	[scope call]
user_data	user data for `func`	

Since: 2.10

---

gtk_print_settings_get_bool ()

```
gboolean
gtk_print_settings_get_bool (GtkPrintSettings *settings,
                             const gchar *key);
```

Returns the boolean represented by the value that is associated with `key` .

The string “true” represents `TRUE`, any other string `FALSE`.

Parameters

settings	a GtkPrintSettings
key	a key

Returns

`TRUE`, if `key` maps to a true value.

Since: 2.10

---

gtk_print_settings_set_bool ()

```
void
gtk_print_settings_set_bool (GtkPrintSettings *settings,
                             const gchar *key,
                             gboolean value);
```

Sets `key` to a boolean value.

Parameters

settings	a GtkPrintSettings
key	a key
value	a boolean

Since: 2.10

---

gtk_print_settings_get_double ()

```
gdouble
gtk_print_settings_get_double (GtkPrintSettings *settings,
                               const gchar *key);
```

Returns the double value associated with `key` , or 0.

Parameters

settings	a GtkPrintSettings
key	a key

Returns

the double value of `key`

Since: 2.10

---

gtk_print_settings_get_double_with_default ()

```
gdouble
gtk_print_settings_get_double_with_default
                                (GtkPrintSettings *settings,
                                 const gchar *key,
                                 gdouble def);
```

Returns the floating point number represented by the value that is associated with `key` , or `default_val` if the value does not represent a floating point number.

Floating point numbers are parsed with `g_ascii_strtod()`.

Parameters

settings	a GtkPrintSettings
key	a key
def	the default value

Returns

the floating point number associated with `key`

Since: 2.10

---

gtk_print_settings_set_double ()

```
void
gtk_print_settings_set_double (GtkPrintSettings *settings,
                               const gchar *key,
                               gdouble value);
```

Sets `key` to a double value.

Parameters

settings	a GtkPrintSettings

key	a key
value	a double value

Since: 2.10

---

gtk_print_settings_get_length ()

```
gdouble
gtk_print_settings_get_length (GtkPrintSettings *settings,
                               const gchar *key,
                               GtkUnit unit);
```

Returns the value associated with `key` , interpreted as a length. The returned value is converted to `units` .

Parameters

settings	a GtkPrintSettings
key	a key
unit	the unit of the return value

Returns

the length value of `key` , converted to `unit`

Since: 2.10

---

gtk_print_settings_set_length ()

```
void
gtk_print_settings_set_length (GtkPrintSettings *settings,
                               const gchar *key,
                               gdouble value,
                               GtkUnit unit);
```

Associates a length in units of `unit` with `key` .

Parameters

settings	a GtkPrintSettings
key	a key
value	a length
unit	the unit of `length`

Since: 2.10

---

gtk_print_settings_get_int ()

```
gint
gtk_print_settings_get_int (GtkPrintSettings *settings,
                            const gchar *key);
```

Returns the integer value of `key` , or 0.

Parameters

settings	a GtkPrintSettings
key	a key

Returns

the integer value of `key`

Since: 2.10

---

gtk_print_settings_get_int_with_default ()

```
gint
gtk_print_settings_get_int_with_default
                                (GtkPrintSettings *settings,
                                 const gchar *key,
                                 gint def);
```

Returns the value of `key` , interpreted as an integer, or the default value.

Parameters

settings	a GtkPrintSettings
key	a key
def	the default value

Returns

the integer value of `key`

Since: 2.10

---

gtk_print_settings_set_int ()

```
void
gtk_print_settings_set_int (GtkPrintSettings *settings,
                            const gchar *key,
                            gint value);
```

Sets `key` to an integer value.

Parameters

settings	a GtkPrintSettings
key	a key

value	an integer

Since: 2.10

---

gtk_print_settings_get_printer ()

```
const gchar *
gtk_print_settings_get_printer (GtkPrintSettings *settings);
```

Convenience function to obtain the value of GTK_PRINT_SETTINGS_PRINTER.

Parameters

settings	a GtkPrintSettings

Returns

the printer name

Since: 2.10

---

gtk_print_settings_set_printer ()

```
void
gtk_print_settings_set_printer (GtkPrintSettings *settings,
                                const gchar *printer);
```

Convenience function to set GTK_PRINT_SETTINGS_PRINTER to `printer` .

Parameters

settings	a GtkPrintSettings
printer	the printer name

Since: 2.10

---

gtk_print_settings_get_orientation ()

```
GtkPageOrientation
gtk_print_settings_get_orientation (GtkPrintSettings *settings);
```

Get the value of GTK_PRINT_SETTINGS_ORIENTATION, converted to a GtkPageOrientation.

Parameters

settings	a GtkPrintSettings

Returns

the orientation

Since: 2.10

---

gtk_print_settings_set_orientation ()

```
void
gtk_print_settings_set_orientation (GtkPrintSettings *settings,
                                    GtkPageOrientation orientation);
```

Sets the value of GTK_PRINT_SETTINGS_ORIENTATION.

Parameters

settings	a GtkPrintSettings
orientation	a page orientation

Since: 2.10

---

gtk_print_settings_get_paper_size ()

```
GtkPaperSize *
gtk_print_settings_get_paper_size (GtkPrintSettings *settings);
```

Gets the value of GTK_PRINT_SETTINGS_PAPER_FORMAT, converted to a GtkPaperSize.

Parameters

settings	a GtkPrintSettings

Returns

the paper size

Since: 2.10

---

gtk_print_settings_set_paper_size ()

```
void
gtk_print_settings_set_paper_size (GtkPrintSettings *settings,
                                   GtkPaperSize *paper_size);
```

Sets the value of GTK_PRINT_SETTINGS_PAPER_FORMAT, GTK_PRINT_SETTINGS_PAPER_WIDTH and GTK_PRINT_SETTINGS_PAPER_HEIGHT.

Parameters

settings	a GtkPrintSettings
paper_size	a paper size

Since: 2.10

---

gtk_print_settings_get_paper_width ()

```
gdouble
gtk_print_settings_get_paper_width (GtkPrintSettings *settings,
                                    GtkUnit unit);
```

Gets the value of GTK_PRINT_SETTINGS_PAPER_WIDTH, converted to unit .

Parameters

settings	a GtkPrintSettings
unit	the unit for the return value

Returns

the paper width, in units of unit

Since: 2.10

---

gtk_print_settings_set_paper_width ()

```
void
gtk_print_settings_set_paper_width (GtkPrintSettings *settings,
                                    gdouble width,
                                    GtkUnit unit);
```

Sets the value of GTK_PRINT_SETTINGS_PAPER_WIDTH.

Parameters

settings	a GtkPrintSettings
width	the paper width
unit	the units of width

Since: 2.10

---

gtk_print_settings_get_paper_height ()

```
gdouble
gtk_print_settings_get_paper_height (GtkPrintSettings *settings,
                                     GtkUnit unit);
```

Gets the value of GTK_PRINT_SETTINGS_PAPER_HEIGHT, converted to unit .

Parameters

settings	a GtkPrintSettings
unit	the unit for the return value

Returns

the paper height, in units of `unit`

Since: 2.10

---

gtk_print_settings_set_paper_height ()

```
void
gtk_print_settings_set_paper_height (GtkPrintSettings *settings,
                                     gdouble height,
                                     GtkUnit unit);
```

Sets the value of `GTK_PRINT_SETTINGS_PAPER_HEIGHT`.

Parameters

settings	a GtkPrintSettings
height	the paper height
unit	the units of `height`

Since: 2.10

---

gtk_print_settings_get_use_color ()

```
gboolean
gtk_print_settings_get_use_color (GtkPrintSettings *settings);
```

Gets the value of `GTK_PRINT_SETTINGS_USE_COLOR`.

Parameters

settings	a GtkPrintSettings

Returns

whether to use color

Since: 2.10

---

gtk_print_settings_set_use_color ()

```
void
gtk_print_settings_set_use_color (GtkPrintSettings *settings,
```

```
                                gboolean use_color);
```
Sets the value of GTK_PRINT_SETTINGS_USE_COLOR.

Parameters

settings	a GtkPrintSettings
use_color	whether to use color

Since: 2.10

---

gtk_print_settings_get_collate ()

```
gboolean
gtk_print_settings_get_collate (GtkPrintSettings *settings);
```
Gets the value of GTK_PRINT_SETTINGS_COLLATE.

Parameters

settings	a GtkPrintSettings

Returns

whether to collate the printed pages

Since: 2.10

---

gtk_print_settings_set_collate ()

```
void
gtk_print_settings_set_collate (GtkPrintSettings *settings,
                                gboolean collate);
```
Sets the value of GTK_PRINT_SETTINGS_COLLATE.

Parameters

settings	a GtkPrintSettings
collate	whether to collate the output

Since: 2.10

---

gtk_print_settings_get_reverse ()

```
gboolean
gtk_print_settings_get_reverse (GtkPrintSettings *settings);
```
Gets the value of GTK_PRINT_SETTINGS_REVERSE.

Parameters

settings	a GtkPrintSettings

Returns

whether to reverse the order of the printed pages

Since: 2.10

---

gtk_print_settings_set_reverse ()

```
void
gtk_print_settings_set_reverse (GtkPrintSettings *settings,
                                gboolean reverse);
```

Sets the value of GTK_PRINT_SETTINGS_REVERSE.

Parameters

settings	a GtkPrintSettings
reverse	whether to reverse the output

Since: 2.10

---

gtk_print_settings_get_duplex ()

```
GtkPrintDuplex
gtk_print_settings_get_duplex (GtkPrintSettings *settings);
```

Gets the value of GTK_PRINT_SETTINGS_DUPLEX.

Parameters

settings	a GtkPrintSettings

Returns

whether to print the output in duplex.

Since: 2.10

---

gtk_print_settings_set_duplex ()

```
void
gtk_print_settings_set_duplex (GtkPrintSettings *settings,
                               GtkPrintDuplex duplex);
```

Sets the value of GTK_PRINT_SETTINGS_DUPLEX.

Parameters

settings	a GtkPrintSettings
duplex	a GtkPrintDuplex value

Since: 2.10

---

gtk_print_settings_get_quality ()

```
GtkPrintQuality
gtk_print_settings_get_quality (GtkPrintSettings *settings);
```

Gets the value of GTK_PRINT_SETTINGS_QUALITY.

Parameters

settings	a GtkPrintSettings

Returns

the print quality

Since: 2.10

---

gtk_print_settings_set_quality ()

```
void
gtk_print_settings_set_quality (GtkPrintSettings *settings,
                                GtkPrintQuality quality);
```

Sets the value of GTK_PRINT_SETTINGS_QUALITY.

Parameters

settings	a GtkPrintSettings
quality	a GtkPrintQuality value

Since: 2.10

---

gtk_print_settings_get_n_copies ()

```
gint
gtk_print_settings_get_n_copies (GtkPrintSettings *settings);
```

Gets the value of GTK_PRINT_SETTINGS_N_COPIES.

Parameters

settings	a GtkPrintSettings

Returns

the number of copies to print

Since: 2.10

---

gtk_print_settings_set_n_copies ()

```
void
gtk_print_settings_set_n_copies (GtkPrintSettings *settings,
                                 gint num_copies);
```

Sets the value of GTK_PRINT_SETTINGS_N_COPIES.

Parameters

settings	a GtkPrintSettings
num_copies	the number of copies

Since: 2.10

---

gtk_print_settings_get_number_up ()

```
gint
gtk_print_settings_get_number_up (GtkPrintSettings *settings);
```

Gets the value of GTK_PRINT_SETTINGS_NUMBER_UP.

Parameters

settings	a GtkPrintSettings

Returns

the number of pages per sheet

Since: 2.10

---

gtk_print_settings_set_number_up ()

```
void
gtk_print_settings_set_number_up (GtkPrintSettings *settings,
                                  gint number_up);
```

Sets the value of GTK_PRINT_SETTINGS_NUMBER_UP.

Parameters

settings	a GtkPrintSettings
number_up	the number of pages per sheet

Since: 2.10

---

gtk_print_settings_get_number_up_layout ()

```
GtkNumberUpLayout
gtk_print_settings_get_number_up_layout
                                (GtkPrintSettings *settings);
```

Gets the value of GTK_PRINT_SETTINGS_NUMBER_UP_LAYOUT.

Parameters

settings	a GtkPrintSettings

Returns

layout of page in number-up mode

Since: 2.14

---

gtk_print_settings_set_number_up_layout ()

```
void
gtk_print_settings_set_number_up_layout
                                (GtkPrintSettings *settings,
                                 GtkNumberUpLayout number_up_layout);
```

Sets the value of GTK_PRINT_SETTINGS_NUMBER_UP_LAYOUT.

Parameters

settings	a GtkPrintSettings
number_up_layout	a GtkNumberUpLayout value

Since: 2.14

---

gtk_print_settings_get_resolution ()

```
gint
gtk_print_settings_get_resolution (GtkPrintSettings *settings);
```

Gets the value of GTK_PRINT_SETTINGS_RESOLUTION.

Parameters

settings	a GtkPrintSettings

Returns

the resolution in dpi

Since: 2.10

---

gtk_print_settings_set_resolution ()

```
void
gtk_print_settings_set_resolution (GtkPrintSettings *settings,
                                   gint resolution);
```

Sets the values of GTK_PRINT_SETTINGS_RESOLUTION, GTK_PRINT_SETTINGS_RESOLUTION_X and GTK_PRINT_SETTINGS_RESOLUTION_Y.

Parameters

settings	a GtkPrintSettings
resolution	the resolution in dpi

Since: 2.10

---

gtk_print_settings_set_resolution_xy ()

```
void
gtk_print_settings_set_resolution_xy (GtkPrintSettings *settings,
                                      gint resolution_x,
                                      gint resolution_y);
```

Sets the values of GTK_PRINT_SETTINGS_RESOLUTION, GTK_PRINT_SETTINGS_RESOLUTION_X and GTK_PRINT_SETTINGS_RESOLUTION_Y.

Parameters

settings	a GtkPrintSettings
resolution_x	the horizontal resolution in dpi
resolution_y	the vertical resolution in dpi

Since: 2.16

---

gtk_print_settings_get_resolution_x ()

```
gint
gtk_print_settings_get_resolution_x (GtkPrintSettings *settings);
```

Gets the value of GTK_PRINT_SETTINGS_RESOLUTION_X.

Parameters

settings	a GtkPrintSettings

Returns

the horizontal resolution in dpi

Since: 2.16

---

gtk_print_settings_get_resolution_y ()

```
gint
gtk_print_settings_get_resolution_y (GtkPrintSettings *settings);
```

Gets the value of `GTK_PRINT_SETTINGS_RESOLUTION_Y`.

Parameters

settings	a GtkPrintSettings

Returns

the vertical resolution in dpi

Since: 2.16

---

gtk_print_settings_get_printer_lpi ()

```
gdouble
gtk_print_settings_get_printer_lpi (GtkPrintSettings *settings);
```

Gets the value of `GTK_PRINT_SETTINGS_PRINTER_LPI`.

Parameters

settings	a GtkPrintSettings

Returns

the resolution in lpi (lines per inch)

Since: 2.16

---

gtk_print_settings_set_printer_lpi ()

```
void
gtk_print_settings_set_printer_lpi (GtkPrintSettings *settings,
                                    gdouble lpi);
```

Sets the value of GTK_PRINT_SETTINGS_PRINTER_LPI.

Parameters

settings	a GtkPrintSettings
lpi	the resolution in lpi (lines per inch)

Since: 2.16

---

gtk_print_settings_get_scale ()

```
gdouble
gtk_print_settings_get_scale (GtkPrintSettings *settings);
```

Gets the value of GTK_PRINT_SETTINGS_SCALE.

Parameters

settings	a GtkPrintSettings

Returns

the scale in percent

Since: 2.10

---

gtk_print_settings_set_scale ()

```
void
gtk_print_settings_set_scale (GtkPrintSettings *settings,
                              gdouble scale);
```

Sets the value of GTK_PRINT_SETTINGS_SCALE.

Parameters

settings	a GtkPrintSettings
scale	the scale in percent

Since: 2.10

---

gtk_print_settings_get_print_pages ()

```
GtkPrintPages
```

```
gtk_print_settings_get_print_pages (GtkPrintSettings *settings);
```
Gets the value of GTK_PRINT_SETTINGS_PRINT_PAGES.

Parameters

settings	a GtkPrintSettings

Returns

which pages to print

Since: 2.10

---

gtk_print_settings_set_print_pages ()

```
void
gtk_print_settings_set_print_pages (GtkPrintSettings *settings,
                                    GtkPrintPages pages);
```
Sets the value of GTK_PRINT_SETTINGS_PRINT_PAGES.

Parameters

settings	a GtkPrintSettings
pages	a GtkPrintPages value

Since: 2.10

---

gtk_print_settings_get_page_ranges ()

```
GtkPageRange *
gtk_print_settings_get_page_ranges (GtkPrintSettings *settings,
                                    gint *num_ranges);
```
Gets the value of GTK_PRINT_SETTINGS_PAGE_RANGES.

Parameters

settings	a GtkPrintSettings	
num_ranges	return location for the length of the returned array.	[out]

Returns

an array of GtkPageRanges. Use `g_free()` to free the array when it is no longer needed.

[array length=num_ranges][transfer full]

Since: 2.10

---

gtk_print_settings_set_page_ranges ()

```
void
gtk_print_settings_set_page_ranges (GtkPrintSettings *settings,
                                    GtkPageRange *page_ranges,
                                    gint num_ranges);
```

Sets the value of GTK_PRINT_SETTINGS_PAGE_RANGES.

Parameters

settings	a GtkPrintSettings	
page_ranges	an array of GtkPageRanges.	[array length=num_ranges]
num_ranges	the length of `page_ranges`	

Since: 2.10

---

gtk_print_settings_get_page_set ()

```
GtkPageSet
gtk_print_settings_get_page_set (GtkPrintSettings *settings);
```

Gets the value of GTK_PRINT_SETTINGS_PAGE_SET.

Parameters

settings	a GtkPrintSettings

Returns

the set of pages to print

Since: 2.10

---

gtk_print_settings_set_page_set ()

```
void
gtk_print_settings_set_page_set (GtkPrintSettings *settings,
                                 GtkPageSet page_set);
```

Sets the value of GTK_PRINT_SETTINGS_PAGE_SET.

Parameters

settings	a GtkPrintSettings
page_set	a GtkPageSet value

Since: 2.10

---

gtk_print_settings_get_default_source ()

```
const gchar *
gtk_print_settings_get_default_source (GtkPrintSettings *settings);
```

Gets the value of GTK_PRINT_SETTINGS_DEFAULT_SOURCE.

Parameters

settings	a GtkPrintSettings

Returns

the default source

Since: 2.10

---

gtk_print_settings_set_default_source ()

```
void
gtk_print_settings_set_default_source (GtkPrintSettings *settings,
                                       const gchar *default_source);
```

Sets the value of GTK_PRINT_SETTINGS_DEFAULT_SOURCE.

Parameters

settings	a GtkPrintSettings
default_source	the default source

Since: 2.10

---

gtk_print_settings_get_media_type ()

```
const gchar *
gtk_print_settings_get_media_type (GtkPrintSettings *settings);
```

Gets the value of GTK_PRINT_SETTINGS_MEDIA_TYPE.

The set of media types is defined in PWG 5101.1-2002 PWG.

Parameters

settings	a GtkPrintSettings

Returns

the media type

Since: 2.10

---

gtk_print_settings_set_media_type ()

```
void
gtk_print_settings_set_media_type (GtkPrintSettings *settings,
                                   const gchar *media_type);
```

Sets the value of GTK_PRINT_SETTINGS_MEDIA_TYPE.

The set of media types is defined in PWG 5101.1-2002 PWG.

Parameters

settings	a GtkPrintSettings
media_type	the media type

Since: 2.10

---

gtk_print_settings_get_dither ()

```
const gchar *
gtk_print_settings_get_dither (GtkPrintSettings *settings);
```

Gets the value of GTK_PRINT_SETTINGS_DITHER.

Parameters

settings	a GtkPrintSettings

Returns

the dithering that is used

Since: 2.10

---

gtk_print_settings_set_dither ()

```
void
gtk_print_settings_set_dither (GtkPrintSettings *settings,
                               const gchar *dither);
```

Sets the value of GTK_PRINT_SETTINGS_DITHER.

Parameters

settings	a GtkPrintSettings
dither	the dithering that is used

Since: 2.10

---

gtk_print_settings_get_finishings ()

```
const gchar *
gtk_print_settings_get_finishings (GtkPrintSettings *settings);
```

Gets the value of GTK_PRINT_SETTINGS_FINISHINGS.

Parameters

settings	a GtkPrintSettings

Returns

the finishings

Since: 2.10

---

gtk_print_settings_set_finishings ()

```
void
gtk_print_settings_set_finishings (GtkPrintSettings *settings,
                                   const gchar *finishings);
```

Sets the value of GTK_PRINT_SETTINGS_FINISHINGS.

Parameters

settings	a GtkPrintSettings
finishings	the finishings

Since: 2.10

---

gtk_print_settings_get_output_bin ()

```
const gchar *
gtk_print_settings_get_output_bin (GtkPrintSettings *settings);
```

Gets the value of GTK_PRINT_SETTINGS_OUTPUT_BIN.

Parameters

settings	a GtkPrintSettings

Returns

the output bin

Since: 2.10

---

gtk_print_settings_set_output_bin ()

```
void
gtk_print_settings_set_output_bin (GtkPrintSettings *settings,
                                   const gchar *output_bin);
```

Sets the value of GTK_PRINT_SETTINGS_OUTPUT_BIN.

Parameters

settings	a GtkPrintSettings	
output_bin	the output bin	

Since: 2.10

---

gtk_print_settings_new_from_file ()

```
GtkPrintSettings *
gtk_print_settings_new_from_file (const gchar *file_name,
                                  GError **error);
```

Reads the print settings from `file_name` . Returns a new GtkPrintSettings object with the restored settings, or `NULL` if an error occurred. If the file could not be loaded then error is set to either a GFileError or GKeyFileError. See `gtk_print_settings_to_file()`.

Parameters

file_name	the filename to read the settings from.	[type filename]
error	return location for errors, or `NULL`.	[allow-none]

Returns

the restored GtkPrintSettings

Since: 2.12

---

gtk_print_settings_new_from_key_file ()

```
GtkPrintSettings *
gtk_print_settings_new_from_key_file (GKeyFile *key_file,
                                      const gchar *group_name,
                                      GError **error);
```

Reads the print settings from the group `group_name` in `key_file` . Returns a new GtkPrintSettings object with the restored settings, or `NULL` if an error occurred. If the file could not be loaded then error is set to either a GFileError or GKeyFileError.

Parameters

key_file	the GKeyFile to retrieve the settings from	
group_name	the name of the group to use, or `NULL` to use the default “Print Settings”.	[allow-none]
error	return location for errors, or `NULL`.	[allow-none]

Returns

the restored GtkPrintSettings

Since: 2.12

---

gtk_print_settings_new_from_gvariant ()

```
GtkPrintSettings *
gtk_print_settings_new_from_gvariant (GVariant *variant);
```

Deserialize print settings from an a{sv} variant in the format produced by `gtk_print_settings_to_gvariant()`.

Parameters

variant	an a{sv} GVariant

Returns

a new GtkPrintSettings object.

[transfer full]

Since: 3.22

---

gtk_print_settings_load_file ()

```
gboolean
gtk_print_settings_load_file (GtkPrintSettings *settings,
                              const gchar *file_name,
                              GError **error);
```

Reads the print settings from `file_name` . If the file could not be loaded then error is set to either a GFileError or GKeyFileError. See `gtk_print_settings_to_file()`.

Parameters

settings	a GtkPrintSettings

file_name	the filename to read the settings from.	[type filename]
error	return location for errors, or `NULL`.	[allow-none]

Returns

`TRUE` on success

Since: 2.14

---

gtk_print_settings_load_key_file ()

```
gboolean
gtk_print_settings_load_key_file (GtkPrintSettings *settings,
                                  GKeyFile *key_file,
                                  const gchar *group_name,
                                  GError **error);
```

Reads the print settings from the group `group_name` in `key_file` . If the file could not be loaded then error is set to either a GFileError or GKeyFileError.

Parameters

settings	a GtkPrintSettings	
key_file	the GKeyFile to retrieve the settings from	
group_name	the name of the group to use, or `NULL` to use the default “Print Settings”.	[allow-none]
error	return location for errors, or `NULL`.	[allow-none]

Returns

`TRUE` on success

Since: 2.14

---

gtk_print_settings_to_file ()

```
gboolean
gtk_print_settings_to_file (GtkPrintSettings *settings,
                            const gchar *file_name,
                            GError **error);
```

This function saves the print settings from `settings` to `file_name` . If the file could not be loaded then error is set to either a GFileError or GKeyFileError.

Parameters

settings	a GtkPrintSettings	
file_name	the file to save to.	[type filename]
error	return location for errors, or `NULL`.	[allow-none]

Returns

`TRUE` on success

Since: 2.12

---

gtk_print_settings_to_key_file ()

```
void
gtk_print_settings_to_key_file (GtkPrintSettings *settings,
                                GKeyFile *key_file,
                                const gchar *group_name);
```

This function adds the print settings from `settings` to `key_file`.

Parameters

settings	a GtkPrintSettings	
key_file	the GKeyFile to save the print settings to	
group_name	the group to add the settings to in `key_file`, or `NULL` to use the default “Print Settings”.	[nullable]

Since: 2.12

---

gtk_print_settings_to_gvariant ()

```
GVariant *
gtk_print_settings_to_gvariant (GtkPrintSettings *settings);
```

Serialize print settings to an a{sv} variant.

Parameters

settings	a GtkPrintSettings

Returns

a new, floating, GVariant.

[transfer none]

Since: 3.22

Types and Values

GtkPrintSettings

```
typedef struct _GtkPrintSettings GtkPrintSettings;
```

GTK_PRINT_SETTINGS_PRINTER

```
#define GTK_PRINT_SETTINGS_PRINTER          "printer"
```

---

enum GtkPageOrientation

See also `gtk_print_settings_set_orientation()`.

Members

GTK_PAGE_ORIENTATION_PORTRAIT	Portrait mode.
GTK_PAGE_ORIENTATION_LANDSCAPE	Landscape mode.
GTK_PAGE_ORIENTATION_REVERSE_PORTRAIT	Reverse portrait mode.
GTK_PAGE_ORIENTATION_REVERSE_LANDSCAPE	Reverse landscape mode.

GTK_PRINT_SETTINGS_ORIENTATION

```
#define GTK_PRINT_SETTINGS_ORIENTATION      "orientation"
```

GTK_PRINT_SETTINGS_PAPER_FORMAT

```
#define GTK_PRINT_SETTINGS_PAPER_FORMAT     "paper-format"
```

GTK_PRINT_SETTINGS_PAPER_WIDTH

```
#define GTK_PRINT_SETTINGS_PAPER_WIDTH      "paper-width"
```

GTK_PRINT_SETTINGS_PAPER_HEIGHT

```
#define GTK_PRINT_SETTINGS_PAPER_HEIGHT     "paper-height"
```

GTK_PRINT_SETTINGS_USE_COLOR

```
#define GTK_PRINT_SETTINGS_USE_COLOR        "use-color"
```

GTK_PRINT_SETTINGS_COLLATE

```
#define GTK_PRINT_SETTINGS_COLLATE          "collate"
```

GTK_PRINT_SETTINGS_REVERSE

```
#define GTK_PRINT_SETTINGS_REVERSE          "reverse"
```

---

enum GtkPrintDuplex

See also gtk_print_settings_set_duplex().

Members

GTK_PRINT_DUPLEX_SIMPLEX	No duplex.
GTK_PRINT_DUPLEX_HORIZONTAL	Horizontal duplex.
GTK_PRINT_DUPLEX_VERTICAL	Vertical duplex.

GTK_PRINT_SETTINGS_DUPLEX

```
#define GTK_PRINT_SETTINGS_DUPLEX           "duplex"
```

---

enum GtkPrintQuality

See also gtk_print_settings_set_quality().

Members

GTK_PRINT_QUALITY_LOW	Low quality.
GTK_PRINT_QUALITY_NORMAL	Normal quality.
GTK_PRINT_QUALITY_HIGH	High quality.
GTK_PRINT_QUALITY_DRAFT	Draft quality.

GTK_PRINT_SETTINGS_QUALITY

```
#define GTK_PRINT_SETTINGS_QUALITY          "quality"
```

GTK_PRINT_SETTINGS_N_COPIES

```
#define GTK_PRINT_SETTINGS_N_COPIES         "n-copies"
```

GTK_PRINT_SETTINGS_NUMBER_UP

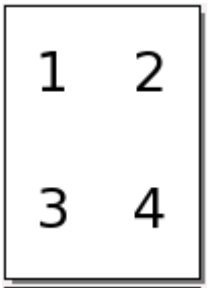
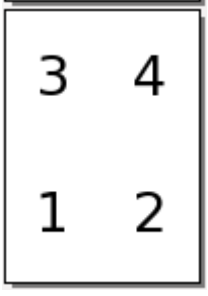
```
#define GTK_PRINT_SETTINGS_NUMBER_UP        "number-up"
```

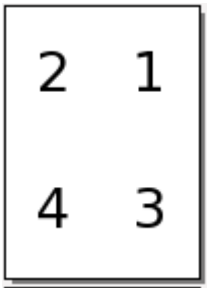
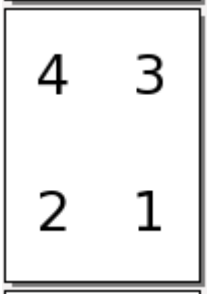
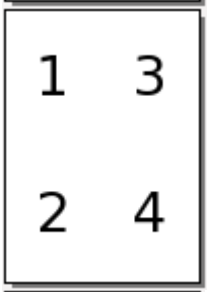
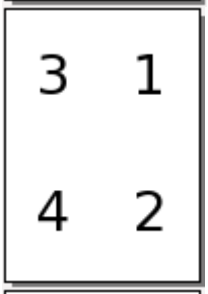
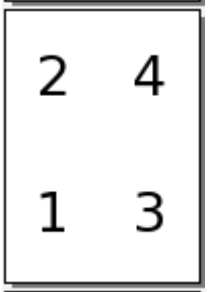
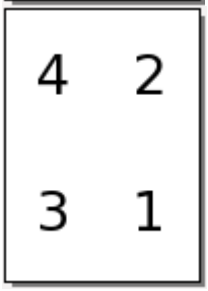
---

enum GtkNumberUpLayout

Used to determine the layout of pages on a sheet when printing multiple pages per sheet.

Members

GTK_NUMBER_UP_LAYOUT_LEFT_TO_RIGHT_TOP_TO_BOTTOM	
GTK_NUMBER_UP_LAYOUT_LEFT_TO_RIGHT_BOTTOM_TO_TOP	

GTK_NUMBER_UP_LAYOUT_RIGHT_TO_LEFT_TOP_TO_BOTTOM	
GTK_NUMBER_UP_LAYOUT_RIGHT_TO_LEFT_BOTTOM_TO_TOP	
GTK_NUMBER_UP_LAYOUT_TOP_TO_BOTTOM_LEFT_TO_RIGHT	
GTK_NUMBER_UP_LAYOUT_TOP_TO_BOTTOM_RIGHT_TO_LEFT	
GTK_NUMBER_UP_LAYOUT_BOTTOM_TO_TOP_LEFT_TO_RIGHT	
GTK_NUMBER_UP_LAYOUT_BOTTOM_TO_TOP_RIGHT_TO_LEFT	

GTK_PRINT_SETTINGS_NUMBER_UP_LAYOUT

```
#define GTK_PRINT_SETTINGS_NUMBER_UP_LAYOUT "number-up-layout"
```

GTK_PRINT_SETTINGS_RESOLUTION

```
#define GTK_PRINT_SETTINGS_RESOLUTION       "resolution"
```

GTK_PRINT_SETTINGS_RESOLUTION_X

```
#define GTK_PRINT_SETTINGS_RESOLUTION_X     "resolution-x"
```

GTK_PRINT_SETTINGS_RESOLUTION_Y

```
#define GTK_PRINT_SETTINGS_RESOLUTION_Y     "resolution-y"
```

GTK_PRINT_SETTINGS_PRINTER_LPI

```
#define GTK_PRINT_SETTINGS_PRINTER_LPI      "printer-lpi"
```

GTK_PRINT_SETTINGS_SCALE

```
#define GTK_PRINT_SETTINGS_SCALE            "scale"
```

enum GtkPrintPages

See also gtk_print_job_set_pages()

Members

GTK_PRINT_PAGES_ALL	All pages.
GTK_PRINT_PAGES_CURRENT	Current page.
GTK_PRINT_PAGES_RANGES	Range of pages.
GTK_PRINT_PAGES_SELECTION	Selected pages.

GTK_PRINT_SETTINGS_PRINT_PAGES

```
#define GTK_PRINT_SETTINGS_PRINT_PAGES      "print-pages"
```

struct GtkPageRange

```
struct GtkPageRange {
  gint start;
  gint end;
};
```

See also gtk_print_settings_set_page_ranges().

Members

gint start;	start of page range.
gint end;	end of page range.

GTK_PRINT_SETTINGS_PAGE_RANGES

```
#define GTK_PRINT_SETTINGS_PAGE_RANGES      "page-ranges"
```

---

enum GtkPageSet

See also gtk_print_job_set_page_set().

Members

GTK_PAGE_SET_ALL	All pages.
GTK_PAGE_SET_EVEN	Even pages.
GTK_PAGE_SET_ODD	Odd pages.

GTK_PRINT_SETTINGS_PAGE_SET

```
#define GTK_PRINT_SETTINGS_PAGE_SET         "page-set"
```

GTK_PRINT_SETTINGS_DEFAULT_SOURCE

```
#define GTK_PRINT_SETTINGS_DEFAULT_SOURCE   "default-source"
```

GTK_PRINT_SETTINGS_MEDIA_TYPE

```
#define GTK_PRINT_SETTINGS_MEDIA_TYPE       "media-type"
```

GTK_PRINT_SETTINGS_DITHER

```
#define GTK_PRINT_SETTINGS_DITHER           "dither"
```

GTK_PRINT_SETTINGS_FINISHINGS

```
#define GTK_PRINT_SETTINGS_FINISHINGS       "finishings"
```

GTK_PRINT_SETTINGS_OUTPUT_BIN

```
#define GTK_PRINT_SETTINGS_OUTPUT_BIN       "output-bin"
```

GTK_PRINT_SETTINGS_OUTPUT_DIR

```
#define GTK_PRINT_SETTINGS_OUTPUT_DIR       "output-dir"
```

The key used by the “Print to file” printer to store the directory to which the output should be written.

Since: 3.6

GTK_PRINT_SETTINGS_OUTPUT_BASENAME

```
#define GTK_PRINT_SETTINGS_OUTPUT_BASENAME  "output-basename"
```

The key used by the “Print to file” printer to store the file name of the output without the path to the directory and the file extension.

Since: 3.6

GTK_PRINT_SETTINGS_OUTPUT_FILE_FORMAT

```
#define GTK_PRINT_SETTINGS_OUTPUT_FILE_FORMAT  "output-file-format"
```

The key used by the “Print to file” printer to store the format of the output. The supported values are “PS” and “PDF”.

GTK_PRINT_SETTINGS_OUTPUT_URI

```
#define GTK_PRINT_SETTINGS_OUTPUT_URI          "output-uri"
```

The key used by the “Print to file” printer to store the URI to which the output should be written. GTK+ itself supports only “file://” URIs.

GTK_PRINT_SETTINGS_WIN32_DRIVER_EXTRA

```
#define GTK_PRINT_SETTINGS_WIN32_DRIVER_EXTRA   "win32-driver-extra"
```

GTK_PRINT_SETTINGS_WIN32_DRIVER_VERSION

```
#define GTK_PRINT_SETTINGS_WIN32_DRIVER_VERSION "win32-driver-version"
```

GtkPageSetup

GtkPageSetup — Stores page setup information

Functions

GtkPageSetup *	gtk_page_setup_new ()
GtkPageSetup *	gtk_page_setup_copy ()
GtkPageOrientation	gtk_page_setup_get_orientation ()
void	gtk_page_setup_set_orientation ()
GtkPaperSize *	gtk_page_setup_get_paper_size ()
void	gtk_page_setup_set_paper_size ()
gdouble	gtk_page_setup_get_top_margin ()
void	gtk_page_setup_set_top_margin ()
gdouble	gtk_page_setup_get_bottom_margin ()
void	gtk_page_setup_set_bottom_margin ()
gdouble	gtk_page_setup_get_left_margin ()
void	gtk_page_setup_set_left_margin ()
gdouble	gtk_page_setup_get_right_margin ()
void	gtk_page_setup_set_right_margin ()
void	gtk_page_setup_set_paper_size_and_default_margins ()
gdouble	gtk_page_setup_get_paper_width ()
gdouble	gtk_page_setup_get_paper_height ()
gdouble	gtk_page_setup_get_page_width ()
gdouble	gtk_page_setup_get_page_height ()
GtkPageSetup *	gtk_page_setup_new_from_file ()
GtkPageSetup *	gtk_page_setup_new_from_key_file ()
GtkPageSetup *	gtk_page_setup_new_from_gvariant ()
gboolean	gtk_page_setup_load_file ()
gboolean	gtk_page_setup_load_key_file ()
gboolean	gtk_page_setup_to_file ()
void	gtk_page_setup_to_key_file ()
GVariant *	gtk_page_setup_to_gvariant ()

Types and Values

GtkPageSetup

Object Hierarchy

```
GObject
└── GtkPageSetup
```

Includes

```
#include <gtk/gtk.h>
```

Description

A GtkPageSetup object stores the page size, orientation and margins. The idea is that you can get one of these from the page setup dialog and then pass it to the GtkPrintOperation when printing. The benefit of splitting this out of the GtkPrintSettings is that these affect the actual layout of the page, and thus need to be set long before user prints.

Margins

The margins specified in this object are the “print margins”, i.e. the parts of the page that the printer cannot print on. These are different from the layout margins that a word processor uses; they are typically used to determine the minimal size for the layout margins.

To obtain a GtkPageSetup use `gtk_page_setup_new()` to get the defaults, or use `gtk_print_run_page_setup_dialog()` to show the page setup dialog and receive the resulting page setup.

A page setup dialog

```
static GtkPrintSettings *settings = NULL;
static GtkPageSetup *page_setup = NULL;

static void
do_page_setup (void)
{
  GtkPageSetup *new_page_setup;

  if (settings == NULL)
    settings = gtk_print_settings_new ();

  new_page_setup =
gtk_print_run_page_setup_dialog (GTK_WINDOW
(main_window),

page_setup, settings);

  if (page_setup)
    g_object_unref (page_setup);

  page_setup = new_page_setup;
}
```

Printing support was added in GTK+ 2.10.

Functions

gtk_page_setup_new ()

```
GtkPageSetup *
gtk_page_setup_new (void);
```

Creates a new GtkPageSetup.

Returns

a new GtkPageSetup.

Since: 2.10

---

gtk_page_setup_copy ()

```
GtkPageSetup *
gtk_page_setup_copy (GtkPageSetup *other);
```

Copies a GtkPageSetup.

Parameters

other	the GtkPageSetup to copy

Returns

a copy of `other`.

[transfer full]

Since: 2.10

---

gtk_page_setup_get_orientation ()

```
GtkPageOrientation
gtk_page_setup_get_orientation (GtkPageSetup *setup);
```

Gets the page orientation of the GtkPageSetup.

Parameters

setup	a GtkPageSetup

Returns

the page orientation

Since: 2.10

---

gtk_page_setup_set_orientation ()

```
void
gtk_page_setup_set_orientation (GtkPageSetup *setup,
                                GtkPageOrientation orientation);
```

Sets the page orientation of the GtkPageSetup.

Parameters

setup	a GtkPageSetup
orientation	a GtkPageOrientation value

Since: 2.10

---

gtk_page_setup_get_paper_size ()

```
GtkPaperSize *
gtk_page_setup_get_paper_size (GtkPageSetup *setup);
```

Gets the paper size of the GtkPageSetup.

Parameters

setup	a GtkPageSetup

Returns

the paper size.

[transfer none]

Since: 2.10

---

gtk_page_setup_set_paper_size ()

```
void
gtk_page_setup_set_paper_size (GtkPageSetup *setup,
                               GtkPaperSize *size);
```

Sets the paper size of the GtkPageSetup without changing the margins. See `gtk_page_setup_set_paper_size_and_default_margins()`.

Parameters

setup	a GtkPageSetup
size	a GtkPaperSize

Since: 2.10

---

gtk_page_setup_get_top_margin ()

```
gdouble
gtk_page_setup_get_top_margin (GtkPageSetup *setup,
                               GtkUnit unit);
```

Gets the top margin in units of `unit` .

Parameters

setup	a GtkPageSetup
unit	the unit for the return value

Returns

the top margin

Since: 2.10

---

gtk_page_setup_set_top_margin ()

```
void
gtk_page_setup_set_top_margin (GtkPageSetup *setup,
                               gdouble margin,
                               GtkUnit unit);
```

Sets the top margin of the GtkPageSetup.

Parameters

setup	a GtkPageSetup
margin	the new top margin in units of `unit`
unit	the units for `margin`

Since: 2.10

---

gtk_page_setup_get_bottom_margin ()

```
gdouble
gtk_page_setup_get_bottom_margin (GtkPageSetup *setup,
                                  GtkUnit unit);
```

Gets the bottom margin in units of `unit` .

Parameters

setup	a GtkPageSetup
unit	the unit for the return value

Returns

the bottom margin

Since: 2.10

---

gtk_page_setup_set_bottom_margin ()

```
void
gtk_page_setup_set_bottom_margin (GtkPageSetup *setup,
                                  gdouble margin,
                                  GtkUnit unit);
```

Sets the bottom margin of the GtkPageSetup.

Parameters

setup	a GtkPageSetup
margin	the new bottom margin in units of `unit`
unit	the units for `margin`

Since: 2.10

---

gtk_page_setup_get_left_margin ()

```
gdouble
gtk_page_setup_get_left_margin (GtkPageSetup *setup,
                                GtkUnit unit);
```

Gets the left margin in units of `unit` .

Parameters

setup	a GtkPageSetup
unit	the unit for the return value

Returns

the left margin

Since: 2.10

---

gtk_page_setup_set_left_margin ()

```
void
gtk_page_setup_set_left_margin (GtkPageSetup *setup,
                                gdouble margin,
                                GtkUnit unit);
```

Sets the left margin of the GtkPageSetup.

Parameters

setup	a GtkPageSetup
margin	the new left margin in units of `unit`
unit	the units for `margin`

Since: 2.10

---

gtk_page_setup_get_right_margin ()

```
gdouble
gtk_page_setup_get_right_margin (GtkPageSetup *setup,
                                 GtkUnit unit);
```

Gets the right margin in units of `unit` .

Parameters

setup	a GtkPageSetup
unit	the unit for the return value

Returns

the right margin

Since: 2.10

---

gtk_page_setup_set_right_margin ()

```
void
gtk_page_setup_set_right_margin (GtkPageSetup *setup,
                                 gdouble margin,
                                 GtkUnit unit);
```

Sets the right margin of the GtkPageSetup.

Parameters

setup	a GtkPageSetup
margin	the new right margin in units of `unit`
unit	the units for `margin`

Since: 2.10

---

gtk_page_setup_set_paper_size_and_default_margins ()

```
void
gtk_page_setup_set_paper_size_and_default_margins
                                (GtkPageSetup *setup,
                                 GtkPaperSize *size);
```

Sets the paper size of the GtkPageSetup and modifies the margins according to the new paper size.

Parameters

setup	a GtkPageSetup
size	a GtkPaperSize

Since: 2.10

---

gtk_page_setup_get_paper_width ()

```
gdouble
gtk_page_setup_get_paper_width (GtkPageSetup *setup,
                                GtkUnit unit);
```

Returns the paper width in units of `unit` .

Note that this function takes orientation, but not margins into consideration. See `gtk_page_setup_get_page_width()`.

Parameters

setup	a GtkPageSetup
unit	the unit for the return value

Returns

the paper width.

Since: 2.10

---

gtk_page_setup_get_paper_height ()

```
gdouble
gtk_page_setup_get_paper_height (GtkPageSetup *setup,
                                 GtkUnit unit);
```

Returns the paper height in units of `unit` .

Note that this function takes orientation, but not margins into consideration. See `gtk_page_setup_get_page_height()`.

Parameters

setup	a GtkPageSetup
unit	the unit for the return value

Returns

the paper height.

Since: 2.10

---

gtk_page_setup_get_page_width ()

```
gdouble
gtk_page_setup_get_page_width (GtkPageSetup *setup,
                               GtkUnit unit);
```

Returns the page width in units of `unit` .

Note that this function takes orientation and margins into consideration. See `gtk_page_setup_get_paper_width()`.

Parameters

setup	a GtkPageSetup
unit	the unit for the return value

Returns

the page width.

Since: 2.10

---

gtk_page_setup_get_page_height ()

```
gdouble
gtk_page_setup_get_page_height (GtkPageSetup *setup,
                                GtkUnit unit);
```

Returns the page height in units of `unit` .

Note that this function takes orientation and margins into consideration. See `gtk_page_setup_get_paper_height()`.

Parameters

setup	a GtkPageSetup
unit	the unit for the return value

Returns

the page height.

Since: 2.10

---

gtk_page_setup_new_from_file ()

```
GtkPageSetup *
gtk_page_setup_new_from_file (const gchar *file_name,
                              GError **error);
```

Reads the page setup from the file `file_name` . Returns a new GtkPageSetup object with the restored page setup, or `NULL` if an error occurred. See `gtk_page_setup_to_file()`.

Parameters

file_name	the filename to read the page setup from.	[type filename]
error	return location for an error, or `NULL`.	[allow-none]

Returns

the restored GtkPageSetup

Since: 2.12

---

gtk_page_setup_new_from_key_file ()

```
GtkPageSetup *
gtk_page_setup_new_from_key_file (GKeyFile *key_file,
                                  const gchar *group_name,
                                  GError **error);
```

Reads the page setup from the group `group_name` in the key file `key_file` . Returns a new GtkPageSetup object with the restored page setup, or `NULL` if an error occurred.

Parameters

key_file	the GKeyFile to retrieve the page_setup from	
group_name	the name of the group in the key_file to read, or `NULL` to use the default name “Page Setup”.	[allow-none]
error	return location for an error, or `NULL`.	[allow-none]

Returns

the restored GtkPageSetup

Since: 2.12

---

gtk_page_setup_new_from_gvariant ()

```
GtkPageSetup *
gtk_page_setup_new_from_gvariant (GVariant *variant);
```

Desrialize a page setup from an a{sv} variant in the format produced by `gtk_page_setup_to_gvariant()`.

Parameters

variant	an a{sv} GVariant

Returns

a new GtkPageSetup object.

[transfer full]

Since: 3.22

---

gtk_page_setup_load_file ()

```
gboolean
gtk_page_setup_load_file (GtkPageSetup *setup,
                          const char *file_name,
                          GError **error);
```

Reads the page setup from the file `file_name` . See `gtk_page_setup_to_file()`.

Parameters

setup	a GtkPageSetup	
file_name	the filename to read the page setup from.	[type filename]
error	return location for an error, or `NULL`.	[allow-none]

Returns

`TRUE` on success

Since: 2.14

---

gtk_page_setup_load_key_file ()

```
gboolean
gtk_page_setup_load_key_file (GtkPageSetup *setup,
                              GKeyFile *key_file,
                              const gchar *group_name,
                              GError **error);
```

Reads the page setup from the group `group_name` in the key file `key_file` .

Parameters

setup	a GtkPageSetup	
key_file	the GKeyFile to retrieve the page_setup from	
group_name	the name of the group in the key_file to read, or `NULL` to use the default name “Page Setup”.	[allow-none]
error	return location for an error, or `NULL`.	[allow-none]

Returns

`TRUE` on success

Since: 2.14

---

gtk_page_setup_to_file ()

```
gboolean
gtk_page_setup_to_file (GtkPageSetup *setup,
                        const char *file_name,
                        GError **error);
```

This function saves the information from `setup` to `file_name` .

Parameters

setup	a GtkPageSetup	
file_name	the file to save to.	[type filename]
error	return location for errors, or `NULL`.	[allow-none]

Returns

`TRUE` on success

Since: 2.12

---

gtk_page_setup_to_key_file ()

```
void
```

```
gtk_page_setup_to_key_file (GtkPageSetup *setup,
                            GKeyFile *key_file,
                            const gchar *group_name);
```

This function adds the page setup from `setup` to `key_file`.

Parameters

setup	a GtkPageSetup	
key_file	the GKeyFile to save the page setup to	
group_name	the group to add the settings to in `key_file`, or `NULL` to use the default name “Page Setup”.	[nullable]

Since: 2.12

---

gtk_page_setup_to_gvariant ()

```
GVariant *
gtk_page_setup_to_gvariant (GtkPageSetup *setup);
```

Serialize page setup to an a{sv} variant.

Return: (transfer none): a new, floating, GVariant

Parameters

setup	a GtkPageSetup

Since: 3.22

Types and Values

GtkPageSetup

```
typedef struct _GtkPageSetup GtkPageSetup;
```

GtkPaperSize

GtkPaperSize — Support for named paper sizes

Functions

GtkPaperSize *	gtk_paper_size_new ()
GtkPaperSize *	gtk_paper_size_new_from_ppd ()
GtkPaperSize *	gtk_paper_size_new_from_ipp ()
GtkPaperSize *	gtk_paper_size_new_custom ()

GtkPaperSize *	gtk_paper_size_copy ()
void	gtk_paper_size_free ()
gboolean	gtk_paper_size_is_equal ()
GList *	gtk_paper_size_get_paper_sizes ()
const gchar *	gtk_paper_size_get_name ()
const gchar *	gtk_paper_size_get_display_name ()
const gchar *	gtk_paper_size_get_ppd_name ()
gdouble	gtk_paper_size_get_width ()
gdouble	gtk_paper_size_get_height ()
gboolean	gtk_paper_size_is_ipp ()
gboolean	gtk_paper_size_is_custom ()
void	gtk_paper_size_set_size ()
gdouble	gtk_paper_size_get_default_top_margin ()
gdouble	gtk_paper_size_get_default_bottom_margin ()
gdouble	gtk_paper_size_get_default_left_margin ()
gdouble	gtk_paper_size_get_default_right_margin ()
const gchar *	gtk_paper_size_get_default ()
GtkPaperSize *	gtk_paper_size_new_from_key_file ()
GtkPaperSize *	gtk_paper_size_new_from_gvariant ()
void	gtk_paper_size_to_key_file ()
GVariant *	gtk_paper_size_to_gvariant ()

Types and Values

	GtkPaperSize
enum	GtkUnit
#define	GTK_UNIT_PIXEL
#define	GTK_PAPER_NAME_A3
#define	GTK_PAPER_NAME_A4
#define	GTK_PAPER_NAME_A5
#define	GTK_PAPER_NAME_B5
#define	GTK_PAPER_NAME_LETTER
#define	GTK_PAPER_NAME_EXECUTIVE
#define	GTK_PAPER_NAME_LEGAL

Object Hierarchy

```
GBoxed
└── GtkPaperSize
```

Includes

```
#include <gtk/gtk.h>
```

Description

GtkPaperSize handles paper sizes. It uses the standard called PWG 5101.1-2002 PWG: Standard for Media Standardized Names to name the paper sizes (and to get the data for the page sizes). In addition to standard paper sizes, GtkPaperSize allows to construct custom paper sizes with arbitrary dimensions.

The GtkPaperSize object stores not only the dimensions (width and height) of a paper size and its name, it also

provides default print margins.

Printing support has been added in GTK+ 2.10.

Functions

gtk_paper_size_new ()

```
GtkPaperSize *
gtk_paper_size_new (const gchar *name);
```

Creates a new GtkPaperSize object by parsing a PWG 5101.1-2002 paper name.

If `name` is `NULL`, the default paper size is returned, see `gtk_paper_size_get_default()`.

Parameters

name	a paper size name, or `NULL`.	[allow-none]

Returns

a new GtkPaperSize, use `gtk_paper_size_free()` to free it

Since: 2.10

---

gtk_paper_size_new_from_ppd ()

```
GtkPaperSize *
gtk_paper_size_new_from_ppd (const gchar *ppd_name,
                             const gchar *ppd_display_name,
                             gdouble width,
                             gdouble height);
```

Creates a new GtkPaperSize object by using PPD information.

If `ppd_name` is not a recognized PPD paper name, `ppd_display_name` , `width` and `height` are used to construct a custom GtkPaperSize object.

Parameters

ppd_name	a PPD paper name
ppd_display_name	the corresponding human-readable name
width	the paper width, in points
height	the paper height in points

Returns

a new GtkPaperSize, use `gtk_paper_size_free()` to free it

Since: 2.10

---

gtk_paper_size_new_from_ipp ()

```
GtkPaperSize *
gtk_paper_size_new_from_ipp (const gchar *ipp_name,
                             gdouble width,
                             gdouble height);
```

Creates a new GtkPaperSize object by using IPP information.

If `ipp_name` is not a recognized paper name, `width` and `height` are used to construct a custom GtkPaperSize object.

Parameters

ipp_name	an IPP paper name
width	the paper width, in points
height	the paper height in points

Returns

a new GtkPaperSize, use `gtk_paper_size_free()` to free it

Since: 3.16

---

gtk_paper_size_new_custom ()

```
GtkPaperSize *
gtk_paper_size_new_custom (const gchar *name,
                           const gchar *display_name,
                           gdouble width,
                           gdouble height,
                           GtkUnit unit);
```

Creates a new GtkPaperSize object with the given parameters.

Parameters

name	the paper name
display_name	the human-readable name
width	the paper width, in units of `unit`
height	the paper height, in units of `unit`
unit	the unit for `width` and `height` . not `GTK_UNIT_NONE`.

Returns

a new GtkPaperSize object, use `gtk_paper_size_free()` to free it

Since: 2.10

---

gtk_paper_size_copy ()

```
GtkPaperSize *
gtk_paper_size_copy (GtkPaperSize *other);
```

Copies an existing GtkPaperSize.

Parameters

other	a GtkPaperSize

Returns

a copy of `other`

Since: 2.10

---

gtk_paper_size_free ()

```
void
gtk_paper_size_free (GtkPaperSize *size);
```

Free the given GtkPaperSize object.

Parameters

size	a GtkPaperSize

Since: 2.10

---

gtk_paper_size_is_equal ()

```
gboolean
gtk_paper_size_is_equal (GtkPaperSize *size1,
                         GtkPaperSize *size2);
```

Compares two GtkPaperSize objects.

Parameters

size1	a GtkPaperSize object
size2	another GtkPaperSize object

Returns

`TRUE`, if `size1` and `size2` represent the same paper size

Since: 2.10

---

gtk_paper_size_get_paper_sizes ()

```
GList *
gtk_paper_size_get_paper_sizes (gboolean include_custom);
```

Creates a list of known paper sizes.

Parameters

include_custom	whether to include custom paper sizes as defined in the page setup dialog

Returns

a newly allocated list of newly allocated GtkPaperSize objects.

[element-type GtkPaperSize][transfer full]

Since: 2.12

---

gtk_paper_size_get_name ()

```
const gchar *
gtk_paper_size_get_name (GtkPaperSize *size);
```

Gets the name of the GtkPaperSize.

Parameters

size	a GtkPaperSize object

Returns

the name of `size`

Since: 2.10

---

gtk_paper_size_get_display_name ()

```
const gchar *
gtk_paper_size_get_display_name (GtkPaperSize *size);
```

Gets the human-readable name of the GtkPaperSize.

Parameters

size	a GtkPaperSize object

Returns

the human-readable name of `size`

Since: 2.10

---

gtk_paper_size_get_ppd_name ()

```
const gchar *
gtk_paper_size_get_ppd_name (GtkPaperSize *size);
```

Gets the PPD name of the GtkPaperSize, which may be `NULL`.

Parameters

size	a GtkPaperSize object

Returns

the PPD name of `size`

Since: 2.10

---

gtk_paper_size_get_width ()

```
gdouble
gtk_paper_size_get_width (GtkPaperSize *size,
                          GtkUnit unit);
```

Gets the paper width of the GtkPaperSize, in units of `unit` .

Parameters

size	a GtkPaperSize object
unit	the unit for the return value, not `GTK_UNIT_NONE`

Returns

the paper width

Since: 2.10

gtk_paper_size_get_height ()

```
gdouble
gtk_paper_size_get_height (GtkPaperSize *size,
                           GtkUnit unit);
```

Gets the paper height of the GtkPaperSize, in units of `unit` .

Parameters

size	a GtkPaperSize object
unit	the unit for the return value, not `GTK_UNIT_NONE`

Returns

the paper height

Since: 2.10

---

gtk_paper_size_is_ipp ()

```
gboolean
gtk_paper_size_is_ipp (GtkPaperSize *size);
```

Returns `TRUE` if `size` is an IPP standard paper size.

Parameters

size	a GtkPaperSize object

Returns

whether `size` is not an IPP custom paper size.

---

gtk_paper_size_is_custom ()

```
gboolean
gtk_paper_size_is_custom (GtkPaperSize *size);
```

Returns `TRUE` if `size` is not a standard paper size.

Parameters

size	a GtkPaperSize object

Returns

whether `size` is a custom paper size.

---

gtk_paper_size_set_size ()

```
void
gtk_paper_size_set_size (GtkPaperSize *size,
                         gdouble width,
                         gdouble height,
                         GtkUnit unit);
```

Changes the dimensions of a `size` to `width` x `height` .

Parameters

size	a custom GtkPaperSize object
width	the new width in units of `unit`
height	the new height in units of `unit`
unit	the unit for `width` and `height`

Since: 2.10

---

gtk_paper_size_get_default_top_margin ()

```
gdouble
gtk_paper_size_get_default_top_margin (GtkPaperSize *size,
                                       GtkUnit unit);
```

Gets the default top margin for the GtkPaperSize.

Parameters

size	a GtkPaperSize object
unit	the unit for the return value, not `GTK_UNIT_NONE`

Returns

the default top margin

Since: 2.10

---

gtk_paper_size_get_default_bottom_margin ()

```
gdouble
gtk_paper_size_get_default_bottom_margin
                                (GtkPaperSize *size,
                                 GtkUnit unit);
```

Gets the default bottom margin for the GtkPaperSize.

Parameters

size	a GtkPaperSize object
unit	the unit for the return value, not GTK_UNIT_NONE

Returns

the default bottom margin

Since: 2.10

---

gtk_paper_size_get_default_left_margin ()

```
gdouble
gtk_paper_size_get_default_left_margin
                                (GtkPaperSize *size,
                                 GtkUnit unit);
```

Gets the default left margin for the GtkPaperSize.

Parameters

size	a GtkPaperSize object
unit	the unit for the return value, not GTK_UNIT_NONE

Returns

the default left margin

Since: 2.10

---

gtk_paper_size_get_default_right_margin ()

```
gdouble
gtk_paper_size_get_default_right_margin
                                (GtkPaperSize *size,
                                 GtkUnit unit);
```

Gets the default right margin for the GtkPaperSize.

Parameters

size	a GtkPaperSize object
unit	the unit for the return value, not GTK_UNIT_NONE

Returns

the default right margin

Since: 2.10

---

gtk_paper_size_get_default ()

```
const gchar *
gtk_paper_size_get_default (void);
```

Returns the name of the default paper size, which depends on the current locale.

Returns

the name of the default paper size. The string is owned by GTK+ and should not be modified.

Since: 2.10

---

gtk_paper_size_new_from_key_file ()

```
GtkPaperSize *
gtk_paper_size_new_from_key_file (GKeyFile *key_file,
                                  const gchar *group_name,
                                  GError **error);
```

Reads a paper size from the group `group_name` in the key file `key_file`.

Parameters

key_file	the GKeyFile to retrieve the papersize from	
group_name	the name of the group in the key file to read, or `NULL` to read the first group.	[nullable]
error	return location for an error, or `NULL`.	[allow-none]

Returns

a new GtkPaperSize object with the restored paper size, or `NULL` if an error occurred

Since: 2.12

---

gtk_paper_size_new_from_gvariant ()

```
GtkPaperSize *
```

```
gtk_paper_size_new_from_gvariant (GVariant *variant);
```
Deserialize a paper size from an a{sv} variant in the format produced by `gtk_paper_size_to_gvariant()`.

Parameters

variant	an a{sv} GVariant

Returns

a new GtkPaperSize object.

[transfer full]

Since: 3.22

---

gtk_paper_size_to_key_file ()

```
void
gtk_paper_size_to_key_file (GtkPaperSize *size,
                            GKeyFile *key_file,
                            const gchar *group_name);
```
This function adds the paper size from `size` to `key_file`.

Parameters

size	a GtkPaperSize
key_file	the GKeyFile to save the paper size to
group_name	the group to add the settings to in `key_file`

Since: 2.12

---

gtk_paper_size_to_gvariant ()

```
GVariant *
gtk_paper_size_to_gvariant (GtkPaperSize *paper_size);
```
Serialize a paper size to an a{sv} variant.

Parameters

paper_size	a GtkPaperSize

Returns

a new, floating, GVariant.

[transfer none]

Since: 3.22

Types and Values

GtkPaperSize

```
typedef struct _GtkPaperSize GtkPaperSize;
```

---

enum GtkUnit

See also `gtk_print_settings_set_paper_width()`.

Members

GTK_UNIT_NONE	No units.
GTK_UNIT_POINTS	Dimensions in points.
GTK_UNIT_INCH	Dimensions in inches.
GTK_UNIT_MM	Dimensions in millimeters

GTK_UNIT_PIXEL

```
#define GTK_UNIT_PIXEL GTK_UNIT_NONE
```

GTK_PAPER_NAME_A3

```
#define GTK_PAPER_NAME_A3 "iso_a3"
```

Name for the A3 paper size.

GTK_PAPER_NAME_A4

```
#define GTK_PAPER_NAME_A4 "iso_a4"
```

Name for the A4 paper size.

GTK_PAPER_NAME_A5

```
#define GTK_PAPER_NAME_A5 "iso_a5"
```

Name for the A5 paper size.

GTK_PAPER_NAME_B5

```
#define GTK_PAPER_NAME_B5 "iso_b5"
```

Name for the B5 paper size.

GTK_PAPER_NAME_LETTER

```
#define GTK_PAPER_NAME_LETTER "na_letter"
```

Name for the Letter paper size.

GTK_PAPER_NAME_EXECUTIVE

```
#define GTK_PAPER_NAME_EXECUTIVE "na_executive"
```

Name for the Executive paper size.

GTK_PAPER_NAME_LEGAL

```
#define GTK_PAPER_NAME_LEGAL "na_legal"
```

Name for the Legal paper size.

See Also

GtkPageSetup

GtkPrinter

GtkPrinter — Represents a printer

Functions

GtkPrinter *	gtk_printer_new ()
GtkPrintBackend *	gtk_printer_get_backend ()
const gchar *	gtk_printer_get_name ()
const gchar *	gtk_printer_get_state_message ()
const gchar *	gtk_printer_get_description ()
const gchar *	gtk_printer_get_location ()
const gchar *	gtk_printer_get_icon_name ()
gint	gtk_printer_get_job_count ()
gboolean	gtk_printer_is_active ()
gboolean	gtk_printer_is_paused ()
gboolean	gtk_printer_is_accepting_jobs ()

gboolean	gtk_printer_is_virtual ()
gboolean	gtk_printer_is_default ()
gboolean	gtk_printer_accepts_ps ()
gboolean	gtk_printer_accepts_pdf ()
GList *	gtk_printer_list_papers ()
gint	gtk_printer_compare ()
gboolean	gtk_printer_has_details ()
void	gtk_printer_request_details ()
GtkPrintCapabilities	gtk_printer_get_capabilities ()
GtkPageSetup *	gtk_printer_get_default_page_size ()
gboolean	gtk_printer_get_hard_margins ()
gboolean	(*GtkPrinterFunc) ()
void	gtk_enumerate_printers ()

Properties

gboolean	accepting-jobs	Read
gboolean	accepts-pdf	Read / Write / Construct Only
gboolean	accepts-ps	Read / Write / Construct Only
GtkPrintBackend *	backend	Read / Write / Construct Only
gchar *	icon-name	Read
gboolean	is-virtual	Read / Write / Construct Only
gint	job-count	Read
gchar *	location	Read
gchar *	name	Read / Write / Construct Only
gboolean	paused	Read
gchar *	state-message	Read

Signals

void	details-acquired	Run Last

Types and Values

struct	GtkPrinter
	GtkPrintBackend

Object Hierarchy

```
GObject
├── GtkPrintBackend
└── GtkPrinter
```

Includes

```
#include <gtk/gtk.h>
```

Description

A GtkPrinter object represents a printer. You only need to deal directly with printers if you use the non-portable GtkPrintUnixDialog API.

A GtkPrinter allows to get status information about the printer, such as its description, its location, the number of queued jobs, etc. Most importantly, a GtkPrinter object can be used to create a GtkPrintJob object, which lets you print to the printer.

Printing support was added in GTK+ 2.10.

Functions

gtk_printer_new ()

```
GtkPrinter *
gtk_printer_new (const gchar *name,
                 GtkPrintBackend *backend,
                 gboolean virtual_);
```

Creates a new GtkPrinter.

Parameters

name	the name of the printer
backend	a GtkPrintBackend
virtual_	whether the printer is virtual

Returns

a new GtkPrinter

Since: 2.10

---

gtk_printer_get_backend ()

```
GtkPrintBackend *
gtk_printer_get_backend (GtkPrinter *printer);
```

Returns the backend of the printer.

Parameters

printer	a GtkPrinter

Returns

the backend of `printer` .

[transfer none]

Since: 2.10

---

gtk_printer_get_name ()

```
const gchar *
gtk_printer_get_name (GtkPrinter *printer);
```

Returns the name of the printer.

Parameters

printer	a GtkPrinter

Returns

the name of `printer`

Since: 2.10

---

gtk_printer_get_state_message ()

```
const gchar *
gtk_printer_get_state_message (GtkPrinter *printer);
```

Returns the state message describing the current state of the printer.

Parameters

printer	a GtkPrinter

Returns

the state message of `printer`

Since: 2.10

---

gtk_printer_get_description ()

```
const gchar *
gtk_printer_get_description (GtkPrinter *printer);
```

Gets the description of the printer.

Parameters

printer	a GtkPrinter

Returns

the description of `printer`

Since: 2.10

---

gtk_printer_get_location ()

```
const gchar *
gtk_printer_get_location (GtkPrinter *printer);
```

Returns a description of the location of the printer.

Parameters

printer	a GtkPrinter

Returns

the location of `printer`

Since: 2.10

---

gtk_printer_get_icon_name ()

```
const gchar *
gtk_printer_get_icon_name (GtkPrinter *printer);
```

Gets the name of the icon to use for the printer.

Parameters

printer	a GtkPrinter

Returns

the icon name for `printer`

Since: 2.10

---

gtk_printer_get_job_count ()

```
gint
gtk_printer_get_job_count (GtkPrinter *printer);
```

Gets the number of jobs currently queued on the printer.

Parameters

printer	a GtkPrinter

Returns

the number of jobs on `printer`

Since: 2.10

---

gtk_printer_is_active ()

```
gboolean
gtk_printer_is_active (GtkPrinter *printer);
```

Returns whether the printer is currently active (i.e. accepts new jobs).

Parameters

printer	a GtkPrinter

Returns

`TRUE` if `printer` is active

Since: 2.10

---

gtk_printer_is_paused ()

```
gboolean
gtk_printer_is_paused (GtkPrinter *printer);
```

Returns whether the printer is currently paused. A paused printer still accepts jobs, but it is not printing them.

Parameters

printer	a GtkPrinter

Returns

`TRUE` if `printer` is paused

Since: 2.14

---

gtk_printer_is_accepting_jobs ()

```
gboolean
```

```
gtk_printer_is_accepting_jobs (GtkPrinter *printer);
```
Returns whether the printer is accepting jobs

Parameters

printer	a GtkPrinter

Returns

`TRUE` if `printer` is accepting jobs

Since: 2.14

---

gtk_printer_is_virtual ()

```
gboolean
gtk_printer_is_virtual (GtkPrinter *printer);
```
Returns whether the printer is virtual (i.e. does not represent actual printer hardware, but something like a CUPS class).

Parameters

printer	a GtkPrinter

Returns

`TRUE` if `printer` is virtual

Since: 2.10

---

gtk_printer_is_default ()

```
gboolean
gtk_printer_is_default (GtkPrinter *printer);
```
Returns whether the printer is the default printer.

Parameters

printer	a GtkPrinter

Returns

`TRUE` if `printer` is the default

Since: 2.10

---

gtk_printer_accepts_ps ()

```
gboolean
gtk_printer_accepts_ps (GtkPrinter *printer);
```

Returns whether the printer accepts input in PostScript format.

Parameters

printer	a GtkPrinter

Returns

`TRUE` if `printer` accepts PostScript

Since: 2.10

---

gtk_printer_accepts_pdf ()

```
gboolean
gtk_printer_accepts_pdf (GtkPrinter *printer);
```

Returns whether the printer accepts input in PDF format.

Parameters

printer	a GtkPrinter

Returns

`TRUE` if `printer` accepts PDF

Since: 2.10

---

gtk_printer_list_papers ()

```
GList *
gtk_printer_list_papers (GtkPrinter *printer);
```

Lists all the paper sizes `printer` supports. This will return and empty list unless the printer's details are available, see `gtk_printer_has_details()` and `gtk_printer_request_details()`.

Parameters

printer	a GtkPrinter

Returns

a newly allocated list of newly allocated GtkPageSetup s.

[element-type GtkPageSetup][transfer full]

Since: 2.12

---

gtk_printer_compare ()

```
gint
gtk_printer_compare (GtkPrinter *a,
                     GtkPrinter *b);
```

Compares two printers.

Parameters

a	a GtkPrinter
b	another GtkPrinter

Returns

0 if the printer match, a negative value if `a` < `b` , or a positive value if `a` > `b`

Since: 2.10

---

gtk_printer_has_details ()

```
gboolean
gtk_printer_has_details (GtkPrinter *printer);
```

Returns whether the printer details are available.

Parameters

printer	a GtkPrinter

Returns

`TRUE` if `printer` details are available

Since: 2.12

---

gtk_printer_request_details ()

```
void
gtk_printer_request_details (GtkPrinter *printer);
```

Requests the printer details. When the details are available, the "details-acquired" signal will be emitted on `printer`.

Parameters

printer	a GtkPrinter

Since: 2.12

---

gtk_printer_get_capabilities ()

```
GtkPrintCapabilities
gtk_printer_get_capabilities (GtkPrinter *printer);
```

Returns the printer's capabilities.

This is useful when you're using GtkPrintUnixDialog's manual-capabilities setting and need to know which settings the printer can handle and which you must handle yourself.

This will return 0 unless the printer's details are available, see `gtk_printer_has_details()` and `gtk_printer_request_details()`.

Parameters

printer	a GtkPrinter

Returns

the printer's capabilities

Since: 2.12

---

gtk_printer_get_default_page_size ()

```
GtkPageSetup *
gtk_printer_get_default_page_size (GtkPrinter *printer);
```

Returns default page size of `printer`.

Parameters

printer	a GtkPrinter

Returns

a newly allocated GtkPageSetup with default page size of the printer.

Since: 2.14

---

gtk_printer_get_hard_margins ()

```
gboolean
gtk_printer_get_hard_margins (GtkPrinter *printer,
                              gdouble *top,
                              gdouble *bottom,
                              gdouble *left,
                              gdouble *right);
```

Retrieve the hard margins of `printer` , i.e. the margins that define the area at the borders of the paper that the printer cannot print to.

Note: This will not succeed unless the printer's details are available, see `gtk_printer_has_details()` and `gtk_printer_request_details()`.

Parameters

printer	a GtkPrinter	
top	a location to store the top margin in.	[out]
bottom	a location to store the bottom margin in.	[out]
left	a location to store the left margin in.	[out]
right	a location to store the right margin in.	[out]

Returns

`TRUE` iff the hard margins were retrieved

Since: 2.20

GtkPrinterFunc ()

```
gboolean
(*GtkPrinterFunc) (GtkPrinter *printer,
                   gpointer data);
```

The type of function passed to `gtk_enumerate_printers()`. Note that you need to ref `printer` , if you want to keep a reference to it after the function has returned.

Parameters

printer	a GtkPrinter	
data	user data passed to `gtk_enumerate_printers()`.	[closure]

Returns

`TRUE` to stop the enumeration, `FALSE` to continue

Since: 2.10

gtk_enumerate_printers ()

```
void
gtk_enumerate_printers (GtkPrinterFunc func,
                        gpointer data,
                        GDestroyNotify destroy,
                        gboolean wait);
```

Calls a function for all GtkPrinters. If `func` returns `TRUE`, the enumeration is stopped.

Parameters

func	a function to call for each printer
data	user data to pass to `func`
destroy	function to call if `data` is no longer needed
wait	if `TRUE`, wait in a recursive mainloop until all printers are enumerated; otherwise return early

Since: 2.10

Types and Values

struct GtkPrinter

```
struct GtkPrinter;
```

GtkPrintBackend

```
typedef struct _GtkPrintBackend GtkPrintBackend;
```

Property Details

The “accepting-jobs” property

```
  “accepting-jobs”           gboolean
```

This property is `TRUE` if the printer is accepting jobs.

Flags: Read

Default value: TRUE

Since: 2.14

The “accepts-pdf” property

“accepts-pdf” gboolean

TRUE if this printer can accept PDF.

Flags: Read / Write / Construct Only

Default value: FALSE

The “accepts-ps” property

“accepts-ps” gboolean

TRUE if this printer can accept PostScript.

Flags: Read / Write / Construct Only

Default value: TRUE

The “backend” property

“backend” GtkPrintBackend *

Backend for the printer.

Flags: Read / Write / Construct Only

The “icon-name” property

“icon-name” gchar *

The icon name to use for the printer.

Flags: Read

Default value: ""

The “is-virtual” property

“is-virtual” gboolean

FALSE if this represents a real hardware printer.

Flags: Read / Write / Construct Only

Default value: FALSE

The “job-count” property

“job-count” gint

Number of jobs queued in the printer.

Flags: Read

Allowed values: >= 0

Default value: 0

The “location” property

“location” gchar *

The location of the printer.

Flags: Read

Default value: ""

The “name” property

“name” gchar *

Name of the printer.

Flags: Read / Write / Construct Only

Default value: ""

The “paused” property

“paused” gboolean

This property is TRUE if this printer is paused. A paused printer still accepts jobs, but it does not print them.

Flags: Read

Default value: FALSE

Since: 2.14

The “state-message” property

“state-message” gchar *

String giving the current state of the printer.

Flags: Read

Default value: ""

Signal Details

The "details-acquired" signal

```
void
user_function (GtkPrinter *printer,
               gboolean    success,
               gpointer    user_data)
```

Gets emitted in response to a request for detailed information about a printer from the print backend. The `success` parameter indicates if the information was actually obtained.

Parameters

printer	the GtkPrinter on which the signal is emitted
success	`TRUE` if the details were successfully acquired
user_data	user data set when the signal handler was connected.

Flags: Run Last

Since: 2.10

GtkPrintJob

GtkPrintJob — Represents a print job

Functions

void	(*GtkPrintJobCompleteFunc) ()
GtkPrintJob *	gtk_print_job_new ()
GtkPrintSettings *	gtk_print_job_get_settings ()
GtkPrinter *	gtk_print_job_get_printer ()
const gchar *	gtk_print_job_get_title ()
GtkPrintStatus	gtk_print_job_get_status ()
gboolean	gtk_print_job_set_source_fd ()
gboolean	gtk_print_job_set_source_file ()
cairo_surface_t *	gtk_print_job_get_surface ()
void	gtk_print_job_send ()
void	gtk_print_job_set_track_print_status ()
gboolean	gtk_print_job_get_track_print_status ()
GtkPrintPages	gtk_print_job_get_pages ()
void	gtk_print_job_set_pages ()
GtkPageRange *	gtk_print_job_get_page_ranges ()
void	gtk_print_job_set_page_ranges ()
GtkPageSet	gtk_print_job_get_page_set ()
void	gtk_print_job_set_page_set ()
gint	gtk_print_job_get_num_copies ()

void	gtk_print_job_set_num_copies ()
gdouble	gtk_print_job_get_scale ()
void	gtk_print_job_set_scale ()
guint	gtk_print_job_get_n_up ()
void	gtk_print_job_set_n_up ()
GtkNumberUpLayout	gtk_print_job_get_n_up_layout ()
void	gtk_print_job_set_n_up_layout ()
gboolean	gtk_print_job_get_rotate ()
void	gtk_print_job_set_rotate ()
gboolean	gtk_print_job_get_collate ()
void	gtk_print_job_set_collate ()
gboolean	gtk_print_job_get_reverse ()
void	gtk_print_job_set_reverse ()

Properties

GtkPageSetup *	page-setup	Read / Write / Construct Only
GtkPrinter *	printer	Read / Write / Construct Only
GtkPrintSettings *	settings	Read / Write / Construct Only
gchar *	title	Read / Write / Construct Only
gboolean	track-print-status	Read / Write

Signals

void	status-changed	Run Last

Types and Values

struct	GtkPrintJob

Object Hierarchy

```
GObject
└── GtkPrintJob
```

Includes

```
#include <gtk/gtkunixprint.h>
```

Description

A GtkPrintJob object represents a job that is sent to a printer. You only need to deal directly with print jobs if you use the non-portable GtkPrintUnixDialog API.

Use `gtk_print_job_get_surface()` to obtain the cairo surface onto which the pages must be drawn. Use `gtk_print_job_send()` to send the finished job to the printer. If you don’t use cairo GtkPrintJob also supports printing of manually generated postscript, via `gtk_print_job_set_source_file()`.

Functions

GtkPrintJobCompleteFunc ()

```
void
(*GtkPrintJobCompleteFunc) (GtkPrintJob *print_job,
                            gpointer user_data,
                            const GError *error);
```

The type of callback that is passed to `gtk_print_job_send()`. It is called when the print job has been completely sent.

Parameters

print_job	the GtkPrintJob
user_data	user data that has been passed to `gtk_print_job_send()`
error	a GError that contains error information if the sending of the print job failed, otherwise `NULL`

---

gtk_print_job_new ()

```
GtkPrintJob *
gtk_print_job_new (const gchar *title,
                   GtkPrinter *printer,
                   GtkPrintSettings *settings,
                   GtkPageSetup *page_setup);
```

Creates a new GtkPrintJob.

Parameters

title	the job title
printer	a GtkPrinter
settings	a GtkPrintSettings
page_setup	a GtkPageSetup

Returns

a new GtkPrintJob

Since: 2.10

---

gtk_print_job_get_settings ()

```
GtkPrintSettings *
gtk_print_job_get_settings (GtkPrintJob *job);
```

Gets the GtkPrintSettings of the print job.

Parameters

job	a GtkPrintJob

Returns

the settings of `job` .

[transfer none]

Since: 2.10

---

gtk_print_job_get_printer ()

```
GtkPrinter *
gtk_print_job_get_printer (GtkPrintJob *job);
```

Gets the GtkPrinter of the print job.

Parameters

job	a GtkPrintJob

Returns

the printer of `job` .

[transfer none]

Since: 2.10

---

gtk_print_job_get_title ()

```
const gchar *
gtk_print_job_get_title (GtkPrintJob *job);
```

Gets the job title.

Parameters

job	a GtkPrintJob

Returns

the title of `job`

Since: 2.10

---

gtk_print_job_get_status ()

```
GtkPrintStatus
gtk_print_job_get_status (GtkPrintJob *job);
```

Gets the status of the print job.

Parameters

job	a GtkPrintJob

Returns

the status of `job`

Since: 2.10

---

gtk_print_job_set_source_fd ()

```
gboolean
gtk_print_job_set_source_fd (GtkPrintJob *job,
                             int fd,
                             GError **error);
```

Make the GtkPrintJob send an existing document to the printing system. The file can be in any format understood by the platforms printing system (typically PostScript, but on many platforms PDF may work too). See `gtk_printer_accepts_pdf()` and `gtk_printer_accepts_ps()`.

This is similar to `gtk_print_job_set_source_file()`, but takes expects an open file descriptor for the file, instead of a filename.

Parameters

job	a GtkPrintJob
fd	a file descriptor
error	return location for errors

Returns

`FALSE` if an error occurred

Since: 3.22

---

gtk_print_job_set_source_file ()

```
gboolean
gtk_print_job_set_source_file (GtkPrintJob *job,
                               const gchar *filename,
```

```
                                GError **error);
```

Make the GtkPrintJob send an existing document to the printing system. The file can be in any format understood by the platforms printing system (typically PostScript, but on many platforms PDF may work too). See `gtk_printer_accepts_pdf()` and `gtk_printer_accepts_ps()`.

Parameters

job	a GtkPrintJob	
filename	the file to be printed.	[type filename]
error	return location for errors	

Returns

`FALSE` if an error occurred

Since: 2.10

---

gtk_print_job_get_surface ()

```
cairo_surface_t *
gtk_print_job_get_surface (GtkPrintJob *job,
                           GError **error);
```

Gets a cairo surface onto which the pages of the print job should be rendered.

Parameters

job	a GtkPrintJob	
error	return location for errors, or `NULL`.	[allow-none]

Returns

the cairo surface of `job` .

[transfer none]

Since: 2.10

---

gtk_print_job_send ()

```
void
gtk_print_job_send (GtkPrintJob *job,
                    GtkPrintJobCompleteFunc callback,
                    gpointer user_data,
                    GDestroyNotify dnotify);
```

Sends the print job off to the printer.

Parameters

job	a GtkPrintJob
callback	function to call when the job completes or an error occurs
user_data	user data that gets passed to `callback`
dnotify	destroy notify for `user_data`

Since: 2.10

---

gtk_print_job_set_track_print_status ()

```
void
gtk_print_job_set_track_print_status (GtkPrintJob *job,
                                      gboolean track_status);
```

If track_status is `TRUE`, the print job will try to continue report on the status of the print job in the printer queues and printer. This can allow your application to show things like “out of paper” issues, and when the print job actually reaches the printer.

This function is often implemented using some form of polling, so it should not be enabled unless needed.

Parameters

job	a GtkPrintJob
track_status	`TRUE` to track status after printing

Since: 2.10

---

gtk_print_job_get_track_print_status ()

```
gboolean
gtk_print_job_get_track_print_status (GtkPrintJob *job);
```

Returns wheter jobs will be tracked after printing. For details, see `gtk_print_job_set_track_print_status()`.

Parameters

job	a GtkPrintJob

Returns

`TRUE` if print job status will be reported after printing

Since: 2.10

---

gtk_print_job_get_pages ()

```
GtkPrintPages
gtk_print_job_get_pages (GtkPrintJob *job);
```

Gets the GtkPrintPages setting for this job.

Parameters

job	a GtkPrintJob	

Returns

the GtkPrintPages setting

Since: 3.0

---

gtk_print_job_set_pages ()

```
void
gtk_print_job_set_pages (GtkPrintJob *job,
                         GtkPrintPages pages);
```

Sets the GtkPrintPages setting for this job.

Parameters

job	a GtkPrintJob	
pages	the GtkPrintPages setting	

Since: 3.0

---

gtk_print_job_get_page_ranges ()

```
GtkPageRange *
gtk_print_job_get_page_ranges (GtkPrintJob *job,
                               gint *n_ranges);
```

Gets the page ranges for this job.

Parameters

job	a GtkPrintJob	
n_ranges	return location for the number of ranges.	[out]

Returns

a pointer to an array of GtkPageRange structs.

[array length=n_ranges][transfer none]

Since: 3.0

---

gtk_print_job_set_page_ranges ()

```
void
gtk_print_job_set_page_ranges (GtkPrintJob *job,
                               GtkPageRange *ranges,
                               gint n_ranges);
```

Sets the page ranges for this job.

Parameters

job	a GtkPrintJob	
ranges	pointer to an array of GtkPageRange structs.	[array length=n_ranges][transfer full]
n_ranges	the length of the `ranges` array	

Since: 3.0

---

gtk_print_job_get_page_set ()

```
GtkPageSet
gtk_print_job_get_page_set (GtkPrintJob *job);
```

Gets the GtkPageSet setting for this job.

Parameters

job	a GtkPrintJob

Returns

the GtkPageSet setting

Since: 3.0

---

gtk_print_job_set_page_set ()

```
void
gtk_print_job_set_page_set (GtkPrintJob *job,
                            GtkPageSet page_set);
```

Sets the GtkPageSet setting for this job.

Parameters

job	a GtkPrintJob

page_set	a GtkPageSet setting

Since: 3.0

---

gtk_print_job_get_num_copies ()

```
gint
gtk_print_job_get_num_copies (GtkPrintJob *job);
```

Gets the number of copies of this job.

Parameters

job	a GtkPrintJob

Returns

the number of copies

Since: 3.0

---

gtk_print_job_set_num_copies ()

```
void
gtk_print_job_set_num_copies (GtkPrintJob *job,
                              gint num_copies);
```

Sets the number of copies for this job.

Parameters

job	a GtkPrintJob
num_copies	the number of copies

Since: 3.0

---

gtk_print_job_get_scale ()

```
gdouble
gtk_print_job_get_scale (GtkPrintJob *job);
```

Gets the scale for this job (where 1.0 means unscaled).

Parameters

job	a GtkPrintJob

Returns

the scale

Since: 3.0

---

gtk_print_job_set_scale ()

```
void
gtk_print_job_set_scale (GtkPrintJob *job,
                         gdouble scale);
```

Sets the scale for this job (where 1.0 means unscaled).

Parameters

job	a GtkPrintJob
scale	the scale

Since: 3.0

---

gtk_print_job_get_n_up ()

```
guint
gtk_print_job_get_n_up (GtkPrintJob *job);
```

Gets the n-up setting for this job.

Parameters

job	a GtkPrintJob

Returns

the n-up setting

Since: 3.0

---

gtk_print_job_set_n_up ()

```
void
gtk_print_job_set_n_up (GtkPrintJob *job,
                        guint n_up);
```

Sets the n-up setting for this job.

Parameters

job	a GtkPrintJob

n_up	the n-up value

Since: 3.0

---

gtk_print_job_get_n_up_layout ()

```
GtkNumberUpLayout
gtk_print_job_get_n_up_layout (GtkPrintJob *job);
```

Gets the n-up layout setting for this job.

Parameters

job	a GtkPrintJob

Returns

the n-up layout

Since: 3.0

---

gtk_print_job_set_n_up_layout ()

```
void
gtk_print_job_set_n_up_layout (GtkPrintJob *job,
                               GtkNumberUpLayout layout);
```

Sets the n-up layout setting for this job.

Parameters

job	a GtkPrintJob
layout	the n-up layout setting

Since: 3.0

---

gtk_print_job_get_rotate ()

```
gboolean
gtk_print_job_get_rotate (GtkPrintJob *job);
```

Gets whether the job is printed rotated.

Parameters

job	a GtkPrintJob

Returns

whether the job is printed rotated

Since: 3.0

---

gtk_print_job_set_rotate ()

```
void
gtk_print_job_set_rotate (GtkPrintJob *job,
                          gboolean rotate);
```

Sets whether this job is printed rotated.

Parameters

job	a GtkPrintJob
rotate	whether to print rotated

Since: 3.0

---

gtk_print_job_get_collate ()

```
gboolean
gtk_print_job_get_collate (GtkPrintJob *job);
```

Gets whether this job is printed collated.

Parameters

job	a GtkPrintJob

Returns

whether the job is printed collated

Since: 3.0

---

gtk_print_job_set_collate ()

```
void
gtk_print_job_set_collate (GtkPrintJob *job,
                           gboolean collate);
```

Sets whether this job is printed collated.

Parameters

job	a GtkPrintJob

collate	whether the job is printed collated

Since: 3.0

---

gtk_print_job_get_reverse ()

```
gboolean
gtk_print_job_get_reverse (GtkPrintJob *job);
```

Gets whether this job is printed reversed.

Parameters

job	a GtkPrintJob

Returns

whether the job is printed reversed.

Since: 3.0

---

gtk_print_job_set_reverse ()

```
void
gtk_print_job_set_reverse (GtkPrintJob *job,
                           gboolean reverse);
```

Sets whether this job is printed reversed.

Parameters

job	a GtkPrintJob
reverse	whether the job is printed reversed

Since: 3.0

Types and Values

struct GtkPrintJob

```
struct GtkPrintJob;
```

Property Details

The “page-setup” property

“page-setup” GtkPageSetup *

Page Setup.

Flags: Read / Write / Construct Only

The “printer” property

“printer” GtkPrinter *

Printer to print the job to.

Flags: Read / Write / Construct Only

The “settings” property

“settings” GtkPrintSettings *

Printer settings.

Flags: Read / Write / Construct Only

The “title” property

“title” gchar *

Title of the print job.

Flags: Read / Write / Construct Only

Default value: NULL

The “track-print-status” property

“track-print-status” gboolean

TRUE if the print job will continue to emit status-changed signals after the print data has been sent to the printer or print server.

Flags: Read / Write

Default value: FALSE

Signal Details

The “status-changed” signal

void

```
user_function (GtkPrintJob *job,
               gpointer     user_data)
```

Gets emitted when the status of a job changes. The signal handler can use `gtk_print_job_get_status()` to obtain the new status.

Parameters

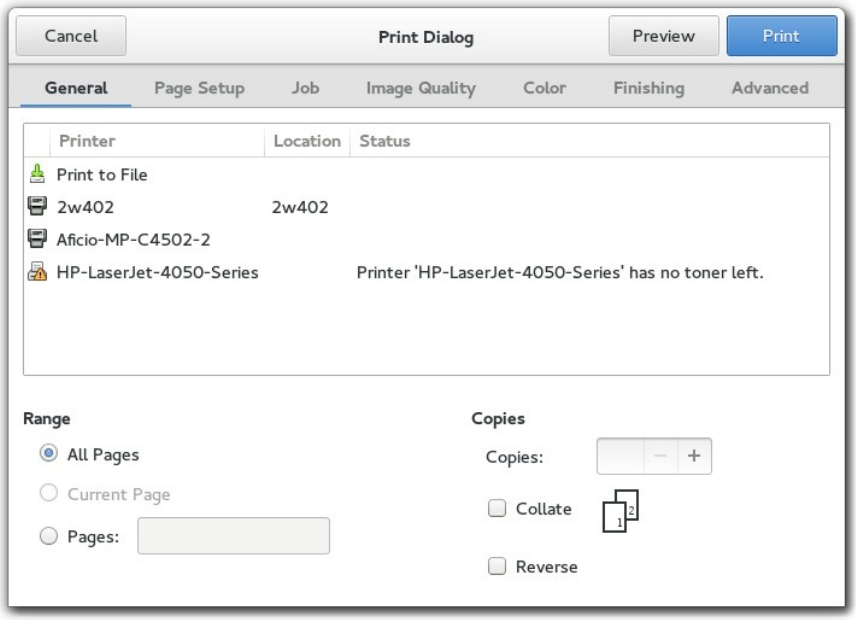
job	the GtkPrintJob object on which the signal was emitted
user_data	user data set when the signal handler was connected.

Flags: Run Last

Since: 2.10

GtkPrintUnixDialog

GtkPrintUnixDialog — A print dialog



Functions

GtkWidget *	gtk_print_unix_dialog_new ()
void	gtk_print_unix_dialog_set_page_setup ()
GtkPageSetup *	gtk_print_unix_dialog_get_page_setup ()
void	gtk_print_unix_dialog_set_current_page ()
gint	gtk_print_unix_dialog_get_current_page ()
void	gtk_print_unix_dialog_set_settings ()
GtkPrintSettings *	gtk_print_unix_dialog_get_settings ()
GtkPrinter *	gtk_print_unix_dialog_get_selected_printer ()
void	gtk_print_unix_dialog_add_custom_tab ()
void	gtk_print_unix_dialog_set_support_selection ()
gboolean	gtk_print_unix_dialog_get_support_selection ()
void	gtk_print_unix_dialog_set_has_selection ()
gboolean	gtk_print_unix_dialog_get_has_selection ()

void	gtk_print_unix_dialog_set_embed_page_setup ()
gboolean	gtk_print_unix_dialog_get_embed_page_setup ()
gboolean	gtk_print_unix_dialog_get_page_setup_set ()
void	gtk_print_unix_dialog_set_manual_capabilities ()
GtkPrintCapabilities	gtk_print_unix_dialog_get_manual_capabilities ()

Properties

gint	current-page	Read / Write
gboolean	embed-page-setup	Read / Write
gboolean	has-selection	Read / Write
GtkPrintCapabilities	manual-capabilities	Read / Write
GtkPageSetup *	page-setup	Read / Write
GtkPrintSettings *	print-settings	Read / Write
GtkPrinter *	selected-printer	Read
gboolean	support-selection	Read / Write

Types and Values

struct	GtkPrintUnixDialog
enum	GtkPrintCapabilities

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkContainer
            └── GtkBin
                └── GtkWindow
                    └── GtkDialog
                        └── GtkPrintUnixDialog
```

Implemented Interfaces

GtkPrintUnixDialog implements AtkImplementorIface and GtkBuildable.

Includes

```
#include <gtk/gtkunixprint.h>
```

Description

GtkPrintUnixDialog implements a print dialog for platforms which don't provide a native print dialog, like Unix. It can be used very much like any other GTK+ dialog, at the cost of the portability offered by the high-level printing API

In order to print something with GtkPrintUnixDialog, you need to use `gtk_print_unix_dialog_get_selected_printer()` to obtain a GtkPrinter object and use it to construct a GtkPrintJob using `gtk_print_job_new()`.

GtkPrintUnixDialog uses the following response values:

- GTK_RESPONSE_OK: for the “Print” button
- GTK_RESPONSE_APPLY: for the “Preview” button
- GTK_RESPONSE_CANCEL: for the “Cancel” button

Printing support was added in GTK+ 2.10.

GtkPrintUnixDialog as GtkBuildable

The GtkPrintUnixDialog implementation of the GtkBuildable interface exposes its `notebook` internal children with the name “notebook”.

An example of a GtkPrintUnixDialog UI definition fragment:

```
<object class="GtkPrintUnixDialog"
id="dialog1">
  <child internal-child="notebook">
    <object class="GtkNotebook"
id="notebook">
      <child>
        <object class="GtkLabel"
id="tabcontent">
        <property name="label">Content on
notebook tab</property>
        </object>
      </child>
      <child type="tab">
        <object class="GtkLabel"
id="tablabel">
          <property name="label">Tab
label</property>
        </object>
        <packing>
          <property name="tab_expand">False</
property>
          <property
name="tab_fill">False</property>
        </packing>
      </child>
    </object>
  </child>
</object>
```

CSS nodes

GtkPrintUnixDialog has a single CSS node with name printdialog.

Functions

gtk_print_unix_dialog_new ()

```
GtkWidget *
gtk_print_unix_dialog_new (const gchar *title,
                           GtkWindow *parent);
```

Creates a new GtkPrintUnixDialog.

Parameters

title	Title of the dialog, or `NULL`.	[allow-none]
parent	Transient parent of the dialog, or `NULL`.	[allow-none]

Returns

a new GtkPrintUnixDialog

Since: 2.10

---

gtk_print_unix_dialog_set_page_setup ()

```
void
gtk_print_unix_dialog_set_page_setup (GtkPrintUnixDialog *dialog,
                                      GtkPageSetup *page_setup);
```

Sets the page setup of the GtkPrintUnixDialog.

Parameters

dialog	a GtkPrintUnixDialog
page_setup	a GtkPageSetup

Since: 2.10

---

gtk_print_unix_dialog_get_page_setup ()

```
GtkPageSetup *
gtk_print_unix_dialog_get_page_setup (GtkPrintUnixDialog *dialog);
```

Gets the page setup that is used by the GtkPrintUnixDialog.

Parameters

dialog	a GtkPrintUnixDialog

Returns

the page setup of `dialog` .

[transfer none]

Since: 2.10

---

gtk_print_unix_dialog_set_current_page ()

```
void
gtk_print_unix_dialog_set_current_page
                                (GtkPrintUnixDialog *dialog,
                                 gint current_page);
```

Sets the current page number. If `current_page` is not -1, this enables the current page choice for the range of pages to print.

Parameters

dialog	a GtkPrintUnixDialog
current_page	the current page number.

Since: 2.10

---

gtk_print_unix_dialog_get_current_page ()

```
gint
gtk_print_unix_dialog_get_current_page
                                (GtkPrintUnixDialog *dialog);
```

Gets the current page of the GtkPrintUnixDialog.

Parameters

dialog	a GtkPrintUnixDialog

Returns

the current page of `dialog`

Since: 2.10

---

gtk_print_unix_dialog_set_settings ()

```
void
gtk_print_unix_dialog_set_settings (GtkPrintUnixDialog *dialog,
                                    GtkPrintSettings *settings);
```

Sets the GtkPrintSettings for the GtkPrintUnixDialog. Typically, this is used to restore saved print settings from a previous print operation before the print dialog is shown.

Parameters

dialog	a GtkPrintUnixDialog	
settings	a GtkPrintSettings, or `NULL`.	[allow-none]

Since: 2.10

---

gtk_print_unix_dialog_get_settings ()

```
GtkPrintSettings *
gtk_print_unix_dialog_get_settings (GtkPrintUnixDialog *dialog);
```

Gets a new GtkPrintSettings object that represents the current values in the print dialog. Note that this creates a new object, and you need to unref it if don't want to keep it.

Parameters

dialog	a GtkPrintUnixDialog

Returns

a new GtkPrintSettings object with the values from `dialog`

Since: 2.10

---

gtk_print_unix_dialog_get_selected_printer ()

```
GtkPrinter *
gtk_print_unix_dialog_get_selected_printer
                                (GtkPrintUnixDialog *dialog);
```

Gets the currently selected printer.

Parameters

dialog	a GtkPrintUnixDialog

Returns

the currently selected printer.

[transfer none]

Since: 2.10

---

gtk_print_unix_dialog_add_custom_tab ()

```
void
gtk_print_unix_dialog_add_custom_tab (GtkPrintUnixDialog *dialog,
```

```
                                    GtkWidget *child,
                                    GtkWidget *tab_label);
```

Adds a custom tab to the print dialog.

Parameters

dialog	a GtkPrintUnixDialog
child	the widget to put in the custom tab
tab_label	the widget to use as tab label

Since: 2.10

---

gtk_print_unix_dialog_set_support_selection ()

```
void
gtk_print_unix_dialog_set_support_selection
                                (GtkPrintUnixDialog *dialog,
                                 gboolean support_selection);
```

Sets whether the print dialog allows user to print a selection.

Parameters

dialog	a GtkPrintUnixDialog
support_selection	`TRUE` to allow print selection

Since: 2.18

---

gtk_print_unix_dialog_get_support_selection ()

```
gboolean
gtk_print_unix_dialog_get_support_selection
                                (GtkPrintUnixDialog *dialog);
```

Gets the value of “support-selection” property.

Parameters

dialog	a GtkPrintUnixDialog

Returns

whether the application supports print of selection

Since: 2.18

---

gtk_print_unix_dialog_set_has_selection ()

```
void
```

```
gtk_print_unix_dialog_set_has_selection
                                (GtkPrintUnixDialog *dialog,
                                 gboolean has_selection);
```

Sets whether a selection exists.

Parameters

dialog	a GtkPrintUnixDialog
has_selection	`TRUE` indicates that a selection exists

Since: 2.18

---

gtk_print_unix_dialog_get_has_selection ()

```
gboolean
gtk_print_unix_dialog_get_has_selection
                                (GtkPrintUnixDialog *dialog);
```

Gets the value of "has-selection" property.

Parameters

dialog	a GtkPrintUnixDialog

Returns

whether there is a selection

Since: 2.18

---

gtk_print_unix_dialog_set_embed_page_setup ()

```
void
gtk_print_unix_dialog_set_embed_page_setup
                                (GtkPrintUnixDialog *dialog,
                                 gboolean embed);
```

Embed page size combo box and orientation combo box into page setup page.

Parameters

dialog	a GtkPrintUnixDialog
embed	embed page setup selection

Since: 2.18

---

gtk_print_unix_dialog_get_embed_page_setup ()

```
gboolean
```

```
gtk_print_unix_dialog_get_embed_page_setup
                                (GtkPrintUnixDialog *dialog);
```

Gets the value of “embed-page-setup” property.

Parameters

dialog	a GtkPrintUnixDialog

Returns

whether there is a selection

Since: 2.18

---

gtk_print_unix_dialog_get_page_setup_set ()

```
gboolean
gtk_print_unix_dialog_get_page_setup_set
                                (GtkPrintUnixDialog *dialog);
```

Gets the page setup that is used by the GtkPrintUnixDialog.

Parameters

dialog	a GtkPrintUnixDialog

Returns

whether a page setup was set by user.

Since: 2.18

---

gtk_print_unix_dialog_set_manual_capabilities ()

```
void
gtk_print_unix_dialog_set_manual_capabilities
                                (GtkPrintUnixDialog *dialog,
                                 GtkPrintCapabilities capabilities);
```

This lets you specify the printing capabilities your application supports. For instance, if you can handle scaling the output then you pass GTK_PRINT_CAPABILITY_SCALE. If you don’t pass that, then the dialog will only let you select the scale if the printing system automatically handles scaling.

Parameters

dialog	a GtkPrintUnixDialog
capabilities	the printing capabilities of your application

Since: 2.10

gtk_print_unix_dialog_get_manual_capabilities ()

```
GtkPrintCapabilities
gtk_print_unix_dialog_get_manual_capabilities
                                (GtkPrintUnixDialog *dialog);
```

Gets the value of “manual-capabilities” property.

Parameters

dialog	a GtkPrintUnixDialog

Returns

the printing capabilities

Since: 2.18

Types and Values

struct GtkPrintUnixDialog

```
struct GtkPrintUnixDialog;
```

enum GtkPrintCapabilities

An enum for specifying which features the print dialog should offer. If neither GTK_PRINT_CAPABILITY_GENERATE_PDF nor GTK_PRINT_CAPABILITY_GENERATE_PS is specified, GTK+ assumes that all formats are supported.

Members

GTK_PRINT_CAPABILITY_PAGE_SET	Print dialog will offer printing even/odd pages.
GTK_PRINT_CAPABILITY_COPIES	Print dialog will allow to print multiple copies.
GTK_PRINT_CAPABILITY_COLLATE	Print dialog will allow to collate multiple copies.
GTK_PRINT_CAPABILITY_REVERSE	Print dialog will allow to print pages in reverse order.
GTK_PRINT_CAPABILITY_SCALE	Print dialog will allow to scale the output.
GTK_PRINT_CAPABILITY_GENERATE_PDF	The program will send the document to the printer in PDF

	format
GTK_PRINT_CAPABILITY_GENERATE_PS	The program will send the document to the printer in Postscript format
GTK_PRINT_CAPABILITY_PREVIEW	Print dialog will offer a preview
GTK_PRINT_CAPABILITY_NUMBER_UP	Print dialog will offer printing multiple pages per sheet. Since 2.12
GTK_PRINT_CAPABILITY_NUMBER_UP_LAYOUT	Print dialog will allow to rearrange pages when printing multiple pages per sheet. Since 2.14

Property Details

The `"current-page"` property

`"current-page"` `gint`

The current page in the document.

Flags: Read / Write

Allowed values: >= -1

Default value: -1

The `"embed-page-setup"` property

`"embed-page-setup"` `gboolean`

TRUE if page setup combos are embedded in GtkPrintUnixDialog.

Flags: Read / Write

Default value: FALSE

The `"has-selection"` property

`"has-selection"` `gboolean`

Whether the application has a selection.

Flags: Read / Write

Default value: FALSE

The `"manual-capabilities"` property

`"manual-capabilities"` `GtkPrintCapabilities`

Capabilities the application can handle.

Flags: Read / Write

The “page-setup” property

“page-setup” GtkPageSetup *

The GtkPageSetup to use.

Flags: Read / Write

The “print-settings” property

“print-settings” GtkPrintSettings *

The GtkPrintSettings used for initializing the dialog.

Flags: Read / Write

The “selected-printer” property

“selected-printer” GtkPrinter *

The GtkPrinter which is selected.

Flags: Read

The “support-selection” property

“support-selection” gboolean

Whether the dialog supports selection.

Flags: Read / Write

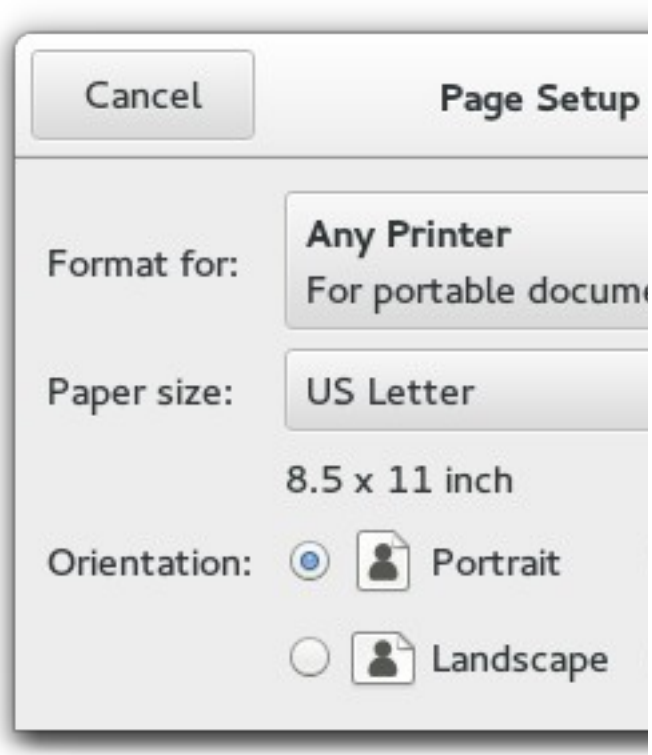
Default value: FALSE

See Also

GtkPageSetupUnixDialog, GtkPrinter, GtkPrintJob

GtkPageSetupUnixDialog

GtkPageSetupUnixDialog — A page setup dialog



Functions

GtkWidget *	gtk_page_setup_unix_dialog_new ()
void	gtk_page_setup_unix_dialog_set_page_setup ()
GtkPageSetup *	gtk_page_setup_unix_dialog_get_page_setup ()
void	gtk_page_setup_unix_dialog_set_print_settings ()
GtkPrintSettings *	gtk_page_setup_unix_dialog_get_print_settings ()

Types and Values

struct	GtkPageSetupUnixDialog
struct	GtkPageSetupUnixDialogClass

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkContainer
            └── GtkBin
                └── GtkWindow
                    └── GtkDialog
                        └── GtkPageSetupUnixDialog
```

Implemented Interfaces

GtkPageSetupUnixDialog implements AtkImplementorIface and GtkBuildable.

Includes

```
#include <gtk/gtkunixprint.h>
```

Description

GtkPageSetupUnixDialog implements a page setup dialog for platforms which don't provide a native page setup dialog, like Unix. It can be used very much like any other GTK+ dialog, at the cost of the portability offered by the high-level printing API

Printing support was added in GTK+ 2.10.

Functions

gtk_page_setup_unix_dialog_new ()

```
GtkWidget *
gtk_page_setup_unix_dialog_new (const gchar *title,
                                GtkWindow *parent);
```

Creates a new page setup dialog.

Parameters

title	the title of the dialog, or `NULL`.	[allow-none]
parent	transient parent of the dialog, or `NULL`.	[allow-none]

Returns

the new GtkPageSetupUnixDialog

Since: 2.10

---

gtk_page_setup_unix_dialog_set_page_setup ()

```
void
gtk_page_setup_unix_dialog_set_page_setup
                               (GtkPageSetupUnixDialog *dialog,
                                GtkPageSetup *page_setup);
```

Sets the GtkPageSetup from which the page setup dialog takes its values.

Parameters

dialog	a GtkPageSetupUnixDialog
page_setup	a GtkPageSetup

Since: 2.10

---

gtk_page_setup_unix_dialog_get_page_setup ()

```
GtkPageSetup *
gtk_page_setup_unix_dialog_get_page_setup
                                (GtkPageSetupUnixDialog *dialog);
```

Gets the currently selected page setup from the dialog.

Parameters

dialog	a GtkPageSetupUnixDialog

Returns

the current page setup.

[transfer none]

Since: 2.10

---

gtk_page_setup_unix_dialog_set_print_settings ()

```
void
gtk_page_setup_unix_dialog_set_print_settings
                                (GtkPageSetupUnixDialog *dialog,
                                 GtkPrintSettings *print_settings);
```

Sets the GtkPrintSettings from which the page setup dialog takes its values.

Parameters

dialog	a GtkPageSetupUnixDialog
print_settings	a GtkPrintSettings

Since: 2.10

---

gtk_page_setup_unix_dialog_get_print_settings ()

```
GtkPrintSettings *
gtk_page_setup_unix_dialog_get_print_settings
                                (GtkPageSetupUnixDialog *dialog);
```

Gets the current print settings from the dialog.

Parameters

dialog	a GtkPageSetupUnixDialog

Returns

the current print settings.

[transfer none]

Since: 2.10

Types and Values

struct GtkPageSetupUnixDialog

```
struct GtkPageSetupUnixDialog;
```

---

struct GtkPageSetupUnixDialogClass

```
struct GtkPageSetupUnixDialogClass {
  GtkDialogClass parent_class;
};
```

Members

Shortcuts Overview

GtkShortcutsWindow — Toplevel which shows help for shortcuts

GtkShortcutsSection — Represents an application mode in a GtkShortcutsWindow

GtkShortcutsGroup — Represents a group of shortcuts in a GtkShortcutsWindow

GtkShortcutsShortcut — Represents a keyboard shortcut in a GtkShortcutsWindow

---

GtkShortcutsWindow

GtkShortcutsWindow — Toplevel which shows help for shortcuts

Properties

gchar *	section-name	Read / Write
gchar *	view-name	Read / Write

Signals

void	close	Action
void	search	Action

Types and Values

struct	GtkShortcutsWindow

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkContainer
            └── GtkBin
                └── GtkWindow
                    └── GtkShortcutsWindow
```

Implemented Interfaces

GtkShortcutsWindow implements AtkImplementorIface and GtkBuildable.

Includes

```
#include <gtk/gtk.h>
```

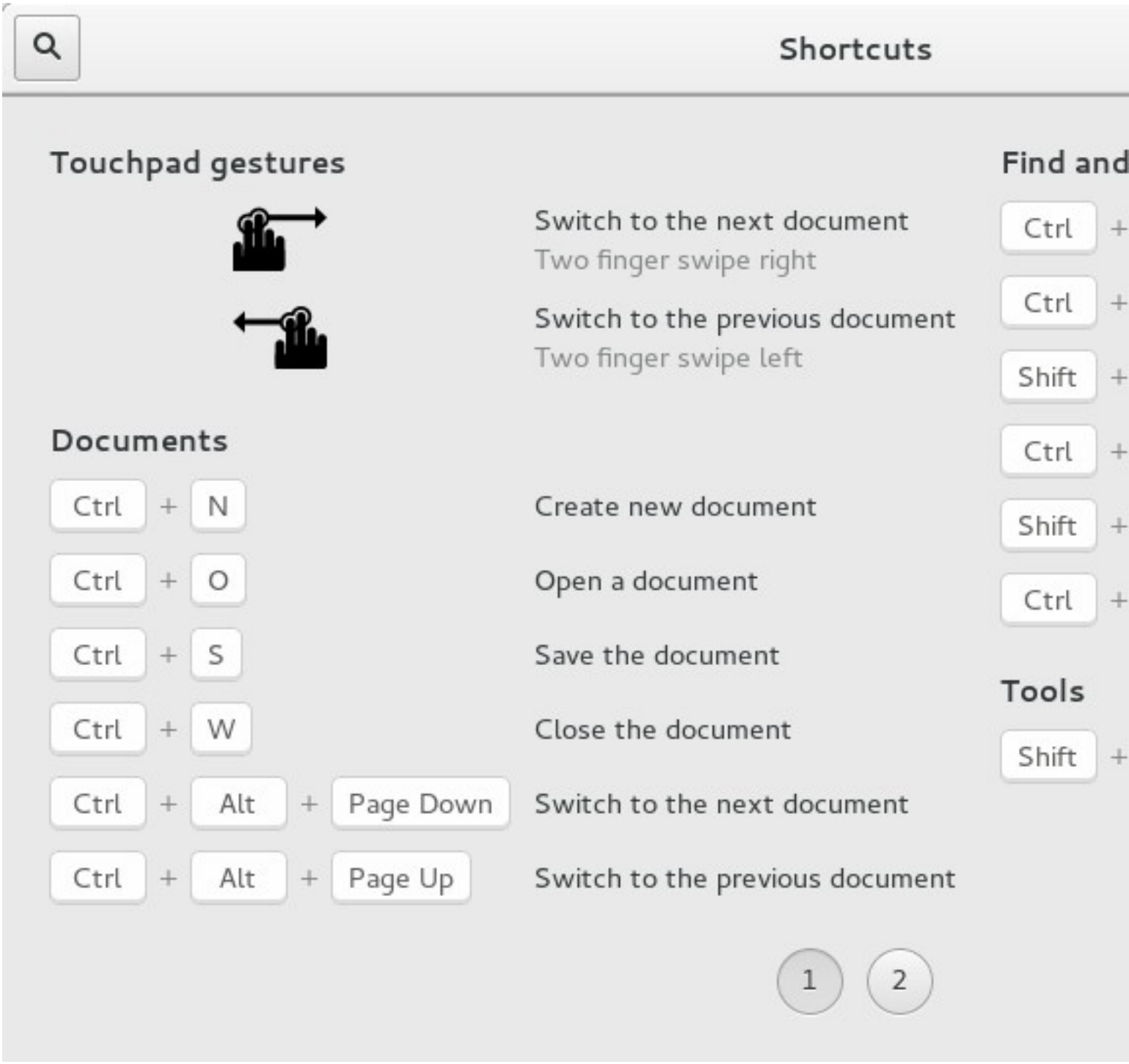
Description

A GtkShortcutsWindow shows brief information about the keyboard shortcuts and gestures of an application. The shortcuts can be grouped, and you can have multiple sections in this window, corresponding to the major modes of your application.

Additionally, the shortcuts can be filtered by the current view, to avoid showing information that is not relevant in the current application context.

The recommended way to construct a GtkShortcutsWindow is with GtkBuilder, by populating a GtkShortcutsWindow with one or more GtkShortcutsSection objects, which contain GtkShortcutsGroups that in turn contain objects of class GtkShortcutsShortcut.

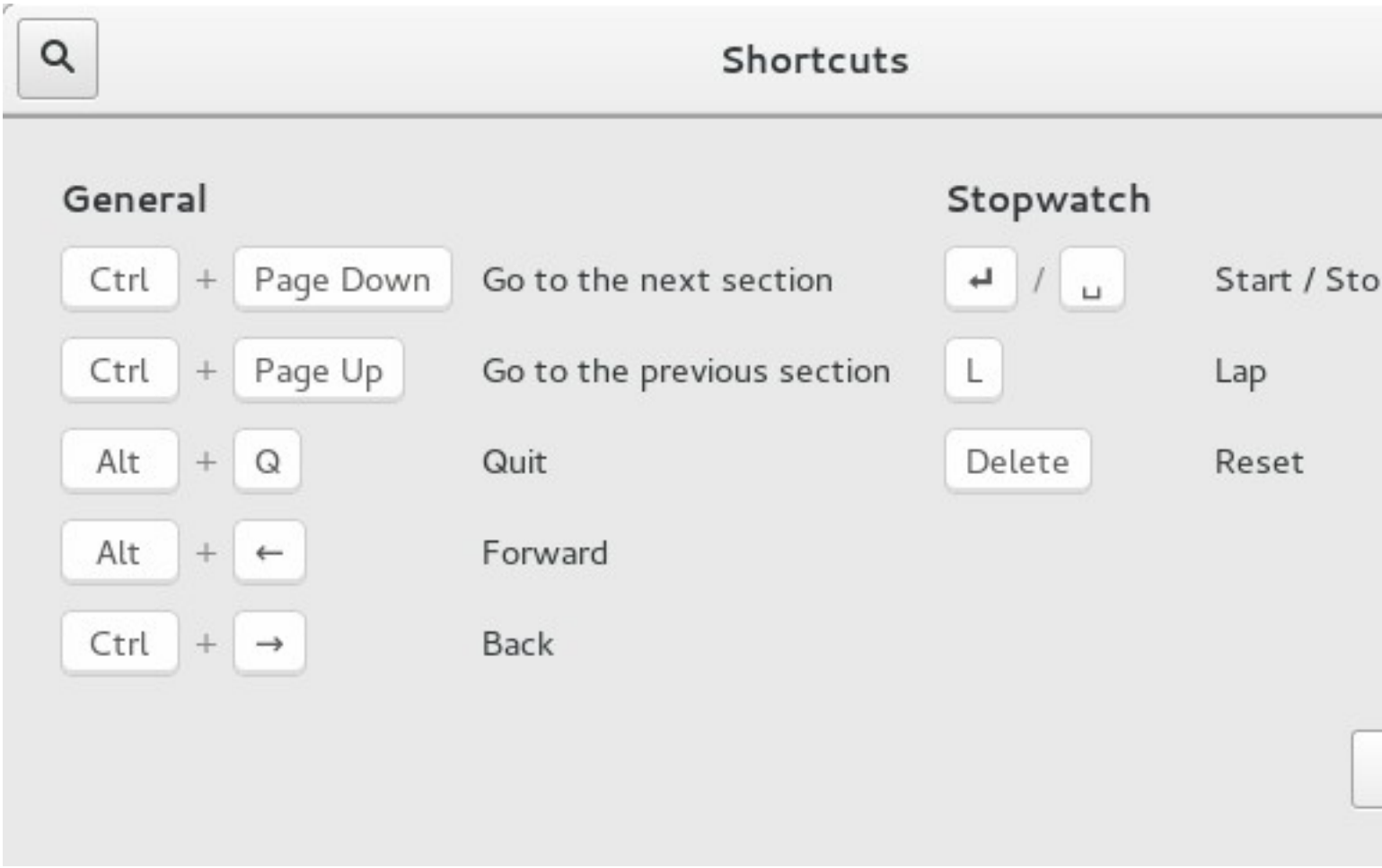
A simple example:



This example has as single section. As you can see, the shortcut groups are arranged in columns, and spread across several pages if there are too many to find on a single page.

The .ui file for this example can be found here.

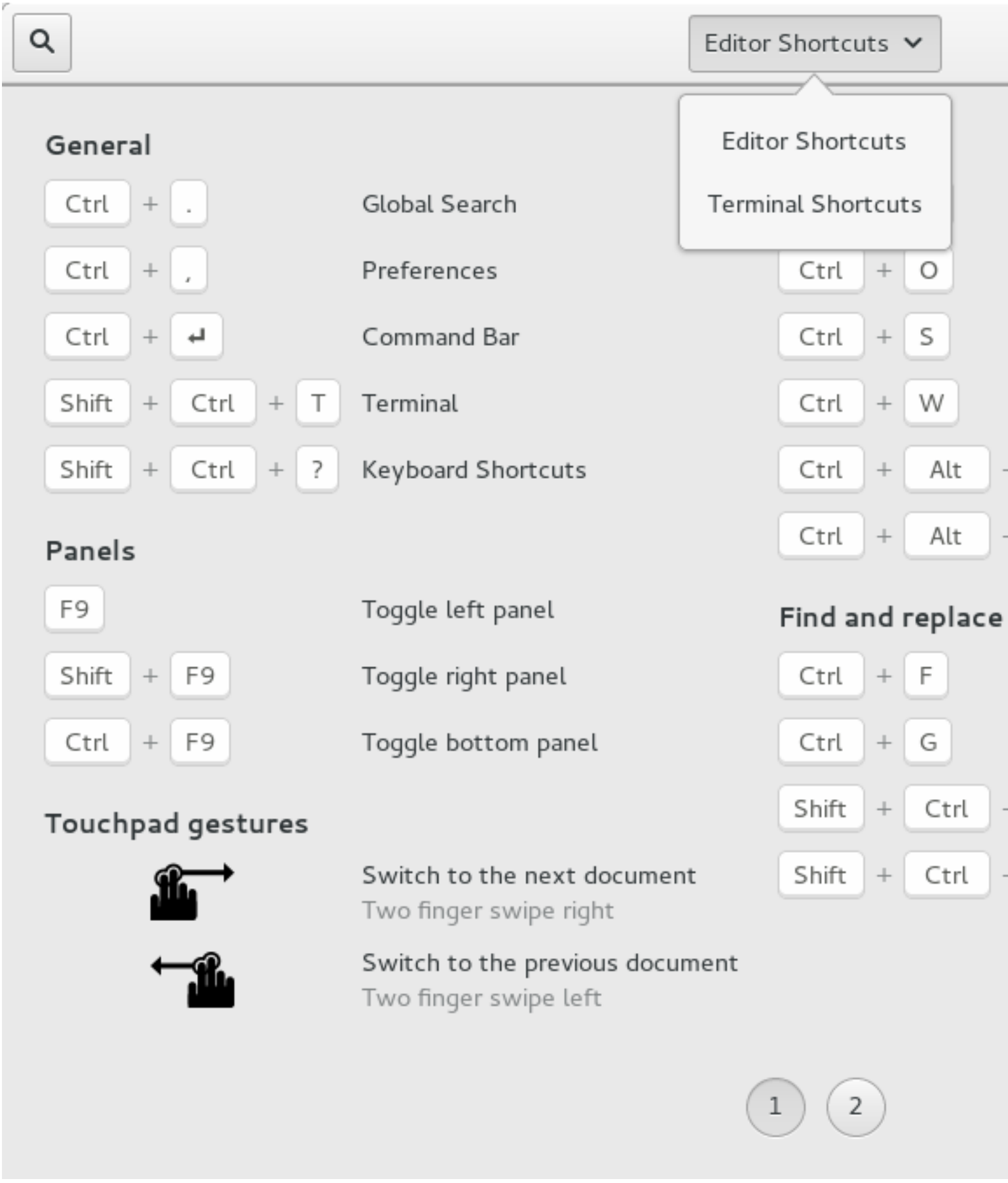
An example with multiple views:



This example shows a GtkShortcutsWindow that has been configured to show only the shortcuts relevant to the "stopwatch" view.

The .ui file for this example can be found here.

An example with multiple sections:



This example shows a GtkShortcutsWindow with two sections, "Editor Shortcuts" and "Terminal Shortcuts".

The .ui file for this example can be found here.

Functions

Types and Values

struct GtkShortcutsWindow

```
struct GtkShortcutsWindow;
```

Property Details

The “section-name” property

```
  “section-name”             gchar *
```

The name of the section to show.

This should be the section-name of one of the GtkShortcutsSection objects that are in this shortcuts window.

Flags: Read / Write

Default value: "internal-search"

The “view-name” property

```
  “view-name”                gchar *
```

The view name by which to filter the contents.

This should correspond to the “view” property of some of the GtkShortcutsGroup objects that are inside this shortcuts window.

Set this to `NULL` to show all groups.

Flags: Read / Write

Default value: NULL

Signal Details

The “close” signal

```
void
user_function (GtkShortcutsWindow *shortcutswindow,
               gpointer            user_data)
```

The ::close signal is a keybinding signal which gets emitted when the user uses a keybinding to close the window.

The default binding for this signal is the Escape key.

Parameters

user_data	user data set when the signal handler was connected.

Flags: Action

The "search" signal

```
void
user_function (GtkShortcutsWindow *shortcutswindow,
               gpointer            user_data)
```

The ::search signal is a keybinding signal which gets emitted when the user uses a keybinding to start a search.

The default binding for this signal is Control-F.

Parameters

user_data	user data set when the signal handler was connected.

Flags: Action

GtkShortcutsSection

GtkShortcutsSection — Represents an application mode in a GtkShortcutsWindow

Properties

guint	max-height	Read / Write
gchar *	section-name	Read / Write
gchar *	title	Read / Write
gchar *	view-name	Read / Write

Signals

gboolean	change-current-page	Action

Types and Values

GtkShortcutsSection

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkContainer
            └── GtkBox
                └── GtkShortcutsSection
```

Implemented Interfaces

GtkShortcutsSection implements AtkImplementorIface, GtkBuildable and GtkOrientable.

Includes

```
#include <gtk/gtk.h>
```

Description

A GtkShortcutsSection collects all the keyboard shortcuts and gestures for a major application mode. If your application needs multiple sections, you should give each section a unique “section-name” and a “title” that can be shown in the section selector of the GtkShortcutsWindow.

The “max-height” property can be used to influence how the groups in the section are distributed over pages and columns.

This widget is only meant to be used with GtkShortcutsWindow.

Functions

Types and Values

GtkShortcutsSection

```
typedef struct _GtkShortcutsSection GtkShortcutsSection;
```

Property Details

The “max-height” property

```
  “max-height”               guint
```

The maximum number of lines to allow per column. This property can be used to influence how the groups in this section are distributed across pages and columns. The default value of 15 should work in most cases.

Flags: Read / Write

Default value: 15

The “section-name” property

```
  “section-name”             gchar *
```

A unique name to identify this section among the sections added to the GtkShortcutsWindow. Setting the “section-name” property to this string will make this section shown in the GtkShortcutsWindow.

Flags: Read / Write

Default value: NULL

The “title” property

```
  “title”                    gchar *
```

The string to show in the section selector of the GtkShortcutsWindow for this section. If there is only one section, you don't need to set a title, since the section selector will not be shown in this case.

Flags: Read / Write

Default value: NULL

The “view-name” property

```
  “view-name”                gchar *
```

A view name to filter the groups in this section by. See “view”.

Applications are expected to use the “view-name” property for this purpose.

Flags: Read / Write

Default value: NULL

Signal Details

The “change-current-page” signal

```
gboolean
user_function (GtkShortcutsSection *shortcutssection,
               gint                 arg1,
               gpointer             user_data)
```

Flags: Action

GtkShortcutsGroup

GtkShortcutsGroup — Represents a group of shortcuts in a GtkShortcutsWindow

Properties

GtkSizeGroup *	accel-size-group	Write
guint	height	Read
gchar *	title	Read / Write
GtkSizeGroup *	title-size-group	Write
gchar *	view	Read / Write

Types and Values

GtkShortcutsGroup

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkContainer
            └── GtkBox
                └── GtkShortcutsGroup
```

Implemented Interfaces

GtkShortcutsGroup implements AtkImplementorIface, GtkBuildable and GtkOrientable.

Includes

```
#include <gtk/gtk.h>
```

Description

A GtkShortcutsGroup represents a group of related keyboard shortcuts or gestures. The group has a title. It may optionally be associated with a view of the application, which can be used to show only relevant shortcuts depending on the application context.

This widget is only meant to be used with GtkShortcutsWindow.

Functions

Types and Values

GtkShortcutsGroup

```
typedef struct _GtkShortcutsGroup GtkShortcutsGroup;
```

Property Details

The “accel-size-group” property

```
“accel-size-group”         GtkSizeGroup *
```

The size group for the accelerator portion of shortcuts in this group.

This is used internally by GTK+, and must not be modified by applications.

Flags: Write

The “height” property

```
“height”                   guint
```

A rough measure for the number of lines in this group.

This is used internally by GTK+, and is not useful for applications.

Flags: Read

Default value: 1

The “title” property

```
“title”                    gchar *
```

The title for this group of shortcuts.

Flags: Read / Write

Default value: ""

The “title-size-group” property

```
“title-size-group”         GtkSizeGroup *
```

The size group for the textual portion of shortcuts in this group.

This is used internally by GTK+, and must not be modified by applications.

Flags: Write

The “view” property

```
“view”                     gchar *
```

An optional view that the shortcuts in this group are relevant for. The group will be hidden if the “view-name” property does not match the view of this group.

Set this to `NULL` to make the group always visible.

Flags: Read / Write

Default value: NULL

GtkShortcutsShortcut

GtkShortcutsShortcut — Represents a keyboard shortcut in a GtkShortcutsWindow

Properties

GtkSizeGroup *	accel-size-group	Write
gchar *	accelerator	Read / Write
gchar *	action-name	Read / Write
GtkTextDirection	direction	Read / Write
GIcon *	icon	Read / Write
gboolean	icon-set	Read / Write
GtkShortcutType	shortcut-type	Read / Write
gchar *	subtitle	Read / Write
gboolean	subtitle-set	Read / Write
gchar *	title	Read / Write
GtkSizeGroup *	title-size-group	Write

Types and Values

	GtkShortcutsShortcut
enum	GtkShortcutType

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkContainer
            └── GtkBox
                └── GtkShortcutsShortcut
```

Implemented Interfaces

GtkShortcutsShortcut implements AtkImplementorIface, GtkBuildable and GtkOrientable.

Includes

```
#include <gtk/gtk.h>
```

Description

A GtkShortcutsShortcut represents a single keyboard shortcut or gesture with a short text. This widget is only meant to be used with GtkShortcutsWindow.

Functions

Types and Values

GtkShortcutsShortcut

```
typedef struct _GtkShortcutsShortcut GtkShortcutsShortcut;
```

---

enum GtkShortcutType

GtkShortcutType specifies the kind of shortcut that is being described. More values may be added to this enumeration over time.

Members

GTK_SHORTCUT_ACCELERATOR	The shortcut is a keyboard accelerator. The "accelerator" property will be used.
GTK_SHORTCUT_GESTURE_PINCH	The shortcut is a pinch gesture. GTK+ provides an icon and subtitle.
GTK_SHORTCUT_GESTURE_STRETCH	The shortcut is a stretch gesture. GTK+ provides an icon and subtitle.
GTK_SHORTCUT_GESTURE_ROTATE_CLOCKWISE	The shortcut is a clockwise rotation gesture. GTK+ provides an icon and subtitle.
GTK_SHORTCUT_GESTURE_ROTATE_COUNTERCLOCKWISE	The shortcut is a counterclockwise rotation gesture. GTK+ provides an icon and subtitle.
GTK_SHORTCUT_GESTURE_TWO_FINGER_SWIPE_LEFT	The shortcut is a two-finger swipe gesture. GTK+ provides an icon and subtitle.
GTK_SHORTCUT_GESTURE_TWO_FINGER_SWIPE_RIGHT	The shortcut is a two-finger swipe gesture. GTK+ provides an icon and subtitle.
GTK_SHORTCUT_GESTURE	The shortcut is a gesture. The "icon" property will be used.

Since: 3.20

Property Details

The “accel-size-group” property

```
  “accel-size-group”         GtkSizeGroup *
```

The size group for the accelerator portion of this shortcut.

This is used internally by GTK+, and must not be modified by applications.

Flags: Write

The “accelerator” property

```
  “accelerator”              gchar *
```

The accelerator(s) represented by this object. This property is used if “shortcut-type” is set to GTK_SHORTCUT_ACCELERATOR.

The syntax of this property is (an extension of) the syntax understood by `gtk_accelerator_parse()`. Multiple accelerators can be specified by separating them with a space, but keep in mind that the available width is limited. It is also possible to specify ranges of shortcuts, using ... between the keys. Sequences of keys can be specified using a + or & between the keys.

Examples:

- A single shortcut: <ctl><alt>delete
- Two alternative shortcuts: <shift>a Home
- A range of shortcuts: <alt>1...<alt>9
- Several keys pressed together: Control_L&Control_R
- A sequence of shortcuts or keys: <ctl>c+<ctl>x

Use + instead of & when the keys may (or have to be) pressed sequentially (e.g use t+t for 'press the t key twice').

Note that <, > and & need to be escaped as <, > and & when used in .ui files.

Flags: Read / Write

Default value: NULL

The “action-name” property

```
  “action-name”              gchar *
```

A detailed action name. If this is set for a shortcut of type `GTK_SHORTCUT_ACCELERATOR`, then GTK+ will use the accelerators that are associated with the action via `gtk_application_set_accels_for_action()`, and setting “accelerator” is not necessary.

Flags: Read / Write

Default value: NULL

Since: 3.22

The “direction” property

“direction” GtkTextDirection

The text direction for which this shortcut is active. If the shortcut is used regardless of the text direction, set this property to GTK_TEXT_DIR_NONE.

Flags: Read / Write

Default value: GTK_TEXT_DIR_NONE

The “icon” property

“icon” GIcon *

An icon to represent the shortcut or gesture. This property is used if “shortcut-type” is set to GTK_SHORTCUT_GESTURE. For the other predefined gesture types, GTK+ provides an icon on its own.

Flags: Read / Write

The “icon-set” property

“icon-set” gboolean

TRUE if an icon has been set.

Flags: Read / Write

Default value: FALSE

The “shortcut-type” property

“shortcut-type” GtkShortcutType

The type of shortcut that is represented.

Flags: Read / Write

Default value: GTK_SHORTCUT_ACCELERATOR

The “subtitle” property

“subtitle” gchar *

The subtitle for the shortcut or gesture.

This is typically used for gestures and should be a short, one-line text that describes the gesture itself. For the predefined gesture types, GTK+ provides a subtitle on its own.

Flags: Read / Write

Default value: ""

The “subtitle-set” property

“subtitle-set” gboolean

TRUE if a subtitle has been set.

Flags: Read / Write

Default value: FALSE

The “title” property

“title” gchar *

The textual description for the shortcut or gesture represented by this object. This should be a short string that can fit in a single line.

Flags: Read / Write

Default value: ""

The “title-size-group” property

“title-size-group” GtkSizeGroup *

The size group for the textual portion of this shortcut.

This is used internally by GTK+, and must not be modified by applications.

Flags: Write

Miscellaneous

GtkAdjustment — A representation of an adjustable bounded value

GtkCalendar — Displays a calendar and allows the user to select a date

GtkDrawingArea — A widget for custom user interface elements

GtkGLArea — A widget for custom drawing with OpenGL

GtkEventBox — A widget used to catch events for widgets which do not have their own window

GtkHandleBox — a widget for detachable window portions

GtkIMContextSimple — An input method context supporting table-based input methods

GtkIMMulticontext — An input method context supporting multiple, loadable input methods

GtkSizeGroup — Grouping widgets so they request the same size

GtkTooltip — Add tips to your widgets

GtkViewport — An adapter which makes widgets scrollable

GtkAccessible — Accessibility support for widgets

GtkAdjustment

GtkAdjustment — A representation of an adjustable bounded value

Functions

GtkAdjustment *	gtk_adjustment_new ()
gdouble	gtk_adjustment_get_value ()
void	gtk_adjustment_set_value ()
void	gtk_adjustment_clamp_page ()
void	gtk_adjustment_changed ()
void	gtk_adjustment_value_changed ()
void	gtk_adjustment_configure ()
gdouble	gtk_adjustment_get_lower ()
gdouble	gtk_adjustment_get_page_increment ()
gdouble	gtk_adjustment_get_page_size ()
gdouble	gtk_adjustment_get_step_increment ()
gdouble	gtk_adjustment_get_minimum_increment ()
gdouble	gtk_adjustment_get_upper ()
void	gtk_adjustment_set_lower ()
void	gtk_adjustment_set_page_increment ()
void	gtk_adjustment_set_page_size ()
void	gtk_adjustment_set_step_increment ()
void	gtk_adjustment_set_upper ()

Properties

gdouble	lower	Read / Write
gdouble	page-increment	Read / Write
gdouble	page-size	Read / Write
gdouble	step-increment	Read / Write
gdouble	upper	Read / Write
gdouble	value	Read / Write

Signals

void	changed	No Recursion
void	value-changed	No Recursion

Types and Values

GtkAdjustment

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkAdjustment
```

Includes

```
#include <gtk/gtk.h>
```

Description

The GtkAdjustment object represents a value which has an associated lower and upper bound, together with step and page increments, and a page size. It is used within several GTK+ widgets, including GtkSpinButton, GtkViewport, and GtkRange (which is a base class for GtkScrollbar and GtkScale).

The GtkAdjustment object does not update the value itself. Instead it is left up to the owner of the GtkAdjustment to control the value.

Functions

gtk_adjustment_new ()

```
GtkAdjustment *
gtk_adjustment_new (gdouble value,
                    gdouble lower,
                    gdouble upper,
                    gdouble step_increment,
                    gdouble page_increment,
                    gdouble page_size);
```

Creates a new GtkAdjustment.

Parameters

value	the initial value
lower	the minimum value
upper	the maximum value
step_increment	the step increment
page_increment	the page increment
page_size	the page size

Returns

a new GtkAdjustment

gtk_adjustment_get_value ()

```
gdouble
gtk_adjustment_get_value (GtkAdjustment *adjustment);
```

Gets the current value of the adjustment. See `gtk_adjustment_set_value()`.

Parameters

adjustment	a GtkAdjustment

Returns

The current value of the adjustment

---

gtk_adjustment_set_value ()

```
void
gtk_adjustment_set_value (GtkAdjustment *adjustment,
                          gdouble value);
```

Sets the GtkAdjustment value. The value is clamped to lie between “lower” and “upper”.

Note that for adjustments which are used in a GtkScrollbar, the effective range of allowed values goes from “lower” to “upper” - “page-size”.

Parameters

adjustment	a GtkAdjustment
value	the new value

---

gtk_adjustment_clamp_page ()

```
void
gtk_adjustment_clamp_page (GtkAdjustment *adjustment,
                           gdouble lower,
                           gdouble upper);
```

Updates the “value” property to ensure that the range between `lower` and `upper` is in the current page (i.e. between “value” and “value” + “page-size”). If the range is larger than the page size, then only the start of it will be in the current page.

A “value-changed” signal will be emitted if the value is changed.

Parameters

adjustment	a GtkAdjustment
lower	the lower value

upper	the upper value

gtk_adjustment_changed ()

```
void
gtk_adjustment_changed (GtkAdjustment *adjustment);
```

`gtk_adjustment_changed` has been deprecated since version 3.18 and should not be used in newly-written code.

GTK+ emits "changed" itself whenever any of the properties (other than value) change

Emits a "changed" signal from the GtkAdjustment. This is typically called by the owner of the GtkAdjustment after it has changed any of the GtkAdjustment properties other than the value.

Parameters

adjustment	a GtkAdjustment

gtk_adjustment_value_changed ()

```
void
gtk_adjustment_value_changed (GtkAdjustment *adjustment);
```

`gtk_adjustment_value_changed` has been deprecated since version 3.18 and should not be used in newly-written code.

GTK+ emits "value-changed" itself whenever the value changes

Emits a "value-changed" signal from the GtkAdjustment. This is typically called by the owner of the GtkAdjustment after it has changed the "value" property.

Parameters

adjustment	a GtkAdjustment

gtk_adjustment_configure ()

```
void
gtk_adjustment_configure (GtkAdjustment *adjustment,
                          gdouble value,
                          gdouble lower,
                          gdouble upper,
                          gdouble step_increment,
                          gdouble page_increment,
                          gdouble page_size);
```

Sets all properties of the adjustment at once.

Use this function to avoid multiple emissions of the "changed" signal. See `gtk_adjustment_set_lower()` for an alternative way of compressing multiple emissions of "changed" into one.

Parameters

adjustment	a GtkAdjustment
value	the new value
lower	the new minimum value
upper	the new maximum value
step_increment	the new step increment
page_increment	the new page increment
page_size	the new page size

Since: 2.14

---

gtk_adjustment_get_lower ()

```
gdouble
gtk_adjustment_get_lower (GtkAdjustment *adjustment);
```

Retrieves the minimum value of the adjustment.

Parameters

adjustment	a GtkAdjustment

Returns

The current minimum value of the adjustment

Since: 2.14

---

gtk_adjustment_get_page_increment ()

```
gdouble
gtk_adjustment_get_page_increment (GtkAdjustment *adjustment);
```

Retrieves the page increment of the adjustment.

Parameters

adjustment	a GtkAdjustment

Returns

The current page increment of the adjustment

Since: 2.14

---

gtk_adjustment_get_page_size ()

```
gdouble
gtk_adjustment_get_page_size (GtkAdjustment *adjustment);
```

Retrieves the page size of the adjustment.

Parameters

adjustment	a GtkAdjustment

Returns

The current page size of the adjustment

Since: 2.14

---

gtk_adjustment_get_step_increment ()

```
gdouble
gtk_adjustment_get_step_increment (GtkAdjustment *adjustment);
```

Retrieves the step increment of the adjustment.

Parameters

adjustment	a GtkAdjustment

Returns

The current step increment of the adjustment.

Since: 2.14

---

gtk_adjustment_get_minimum_increment ()

```
gdouble
gtk_adjustment_get_minimum_increment (GtkAdjustment *adjustment);
```

Gets the smaller of step increment and page increment.

Parameters

adjustment	a GtkAdjustment

Returns

the minimum increment of `adjustment`

Since: 3.2

gtk_adjustment_get_upper ()

```
gdouble
gtk_adjustment_get_upper (GtkAdjustment *adjustment);
```

Retrieves the maximum value of the adjustment.

Parameters

adjustment	a GtkAdjustment

Returns

The current maximum value of the adjustment

Since: 2.14

---

gtk_adjustment_set_lower ()

```
void
gtk_adjustment_set_lower (GtkAdjustment *adjustment,
                          gdouble lower);
```

Sets the minimum value of the adjustment.

When setting multiple adjustment properties via their individual setters, multiple "changed" signals will be emitted. However, since the emission of the "changed" signal is tied to the emission of the "notify" signals of the changed properties, it's possible to compress the "changed" signals into one by calling `g_object_freeze_notify()` and `g_object_thaw_notify()` around the calls to the individual setters.

Alternatively, using a single `g_object_set()` for all the properties to change, or using `gtk_adjustment_configure()` has the same effect of compressing "changed" emissions.

Parameters

adjustment	a GtkAdjustment
lower	the new minimum value

Since: 2.14

---

gtk_adjustment_set_page_increment ()

```
void
gtk_adjustment_set_page_increment (GtkAdjustment *adjustment,
                                   gdouble page_increment);
```

Sets the page increment of the adjustment.

See `gtk_adjustment_set_lower()` about how to compress multiple emissions of the "changed" signal when

setting multiple adjustment properties.

Parameters

adjustment	a GtkAdjustment
page_increment	the new page increment

Since: 2.14

---

gtk_adjustment_set_page_size ()

```
void
gtk_adjustment_set_page_size (GtkAdjustment *adjustment,
                              gdouble page_size);
```

Sets the page size of the adjustment.

See `gtk_adjustment_set_lower()` about how to compress multiple emissions of the GtkAdjustment::changed signal when setting multiple adjustment properties.

Parameters

adjustment	a GtkAdjustment
page_size	the new page size

Since: 2.14

---

gtk_adjustment_set_step_increment ()

```
void
gtk_adjustment_set_step_increment (GtkAdjustment *adjustment,
                                   gdouble step_increment);
```

Sets the step increment of the adjustment.

See `gtk_adjustment_set_lower()` about how to compress multiple emissions of the "changed" signal when setting multiple adjustment properties.

Parameters

adjustment	a GtkAdjustment
step_increment	the new step increment

Since: 2.14

---

gtk_adjustment_set_upper ()

```
void
gtk_adjustment_set_upper (GtkAdjustment *adjustment,
                          gdouble upper);
```

Sets the maximum value of the adjustment.

Note that values will be restricted by `upper - page-size` if the page-size property is nonzero.

See `gtk_adjustment_set_lower()` about how to compress multiple emissions of the "changed" signal when setting multiple adjustment properties.

Parameters

adjustment	a GtkAdjustment
upper	the new maximum value

Since: 2.14

Types and Values

GtkAdjustment

```
typedef struct _GtkAdjustment GtkAdjustment;
```

The GtkAdjustment contains only private fields and should not be directly accessed.

Property Details

The `"lower"` property

`"lower"` `gdouble`

The minimum value of the adjustment.

Flags: Read / Write

Default value: 0

Since: 2.4

The `"page-increment"` property

`"page-increment"` `gdouble`

The page increment of the adjustment.

Flags: Read / Write

Default value: 0

Since: 2.4

The “page-size” property

“page-size” gdouble

The page size of the adjustment. Note that the page-size is irrelevant and should be set to zero if the adjustment is used for a simple scalar value, e.g. in a GtkSpinButton.

Flags: Read / Write

Default value: 0

Since: 2.4

The “step-increment” property

“step-increment” gdouble

The step increment of the adjustment.

Flags: Read / Write

Default value: 0

Since: 2.4

The “upper” property

“upper” gdouble

The maximum value of the adjustment. Note that values will be restricted by `upper - page-size` if the page-size property is nonzero.

Flags: Read / Write

Default value: 0

Since: 2.4

The “value” property

“value” gdouble

The value of the adjustment.

Flags: Read / Write

Default value: 0

Since: 2.4

Signal Details

The “changed” signal

```
void
user_function (GtkAdjustment *adjustment,
               gpointer       user_data)
```

Emitted when one or more of the GtkAdjustment properties have been changed, other than the “value” property.

Parameters

adjustment	the object which received the signal
user_data	user data set when the signal handler was connected.

Flags: No Recursion

The “value-changed” signal

```
void
user_function (GtkAdjustment *adjustment,
               gpointer       user_data)
```

Emitted when the “value” property has been changed.

Parameters

adjustment	the object which received the signal
user_data	user data set when the signal handler was connected.

Flags: No Recursion

GtkCalendar

GtkCalendar — Displays a calendar and allows the user to select a date

Functions

gchar *	(*GtkCalendarDetailFunc) ()
GtkWidget *	gtk_calendar_new ()
void	gtk_calendar_select_month ()
void	gtk_calendar_select_day ()
void	gtk_calendar_mark_day ()
void	gtk_calendar_unmark_day ()
gboolean	gtk_calendar_get_day_is_marked ()
void	gtk_calendar_clear_marks ()
GtkCalendarDisplayOptions	gtk_calendar_get_display_options ()
void	gtk_calendar_set_display_options ()
void	gtk_calendar_get_date ()
void	gtk_calendar_set_detail_func ()

gint	gtk_calendar_get_detail_width_chars ()
void	gtk_calendar_set_detail_width_chars ()
gint	gtk_calendar_get_detail_height_rows ()
void	gtk_calendar_set_detail_height_rows ()

Properties

gint	day	Read / Write
gint	detail-height-rows	Read / Write
gint	detail-width-chars	Read / Write
gint	month	Read / Write
gboolean	no-month-change	Read / Write
gboolean	show-day-names	Read / Write
gboolean	show-details	Read / Write
gboolean	show-heading	Read / Write
gboolean	show-week-numbers	Read / Write
gint	year	Read / Write

Style Properties

gint	horizontal-separation	Read
gint	inner-border	Read
gint	vertical-separation	Read

Signals

void	day-selected	Run First
void	day-selected-double-click	Run First
void	month-changed	Run First
void	next-month	Run First
void	next-year	Run First
void	prev-month	Run First
void	prev-year	Run First

Types and Values

struct	GtkCalendar
enum	GtkCalendarDisplayOptions

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkCalendar
```

Implemented Interfaces

GtkCalendar implements AtkImplementorIface and GtkBuildable.

Includes

```
#include <gtk/gtk.h>
```

Description

GtkCalendar is a widget that displays a Gregorian calendar, one month at a time. It can be created with `gtk_calendar_new()`.

The month and year currently displayed can be altered with `gtk_calendar_select_month()`. The exact day can be selected from the displayed month using `gtk_calendar_select_day()`.

To place a visual marker on a particular day, use `gtk_calendar_mark_day()` and to remove the marker, `gtk_calendar_unmark_day()`. Alternative, all marks can be cleared with `gtk_calendar_clear_marks()`.

The way in which the calendar itself is displayed can be altered using `gtk_calendar_set_display_options()`.

The selected date can be retrieved from a GtkCalendar using `gtk_calendar_get_date()`.

Users should be aware that, although the Gregorian calendar is the legal calendar in most countries, it was adopted progressively between 1582 and 1929. Display before these dates is likely to be historically incorrect.

Functions

GtkCalendarDetailFunc ()

```
gchar *
(*GtkCalendarDetailFunc) (GtkCalendar *calendar,
                          guint year,
                          guint month,
                          guint day,
                          gpointer user_data);
```

This kind of functions provide Pango markup with detail information for the specified day. Examples for such details are holidays or appointments. The function returns `NULL` when no information is available.

Parameters

calendar	a GtkCalendar.
year	the year for which details are needed.
month	the month for which details are needed.
day	the day of `month` for which details are needed.
user_data	the data passed with `gtk_calendar_set_detail_func()`.

Returns

Newly allocated string with Pango markup with details for the specified day or `NULL`.

[nullable][transfer full]

Since: 2.14

---

gtk_calendar_new ()

```
GtkWidget *
gtk_calendar_new (void);
```

Creates a new calendar, with the current date being selected.

Returns

a newly GtkCalendar widget

---

gtk_calendar_select_month ()

```
void
gtk_calendar_select_month (GtkCalendar *calendar,
                           guint month,
                           guint year);
```

Shifts the calendar to a different month.

Parameters

calendar	a GtkCalendar
month	a month number between 0 and 11.
year	the year the month is in.

---

gtk_calendar_select_day ()

```
void
gtk_calendar_select_day (GtkCalendar *calendar,
                         guint day);
```

Selects a day from the current month.

Parameters

calendar	a GtkCalendar.
day	the day number between 1 and 31, or 0 to unselect the currently selected day.

---

gtk_calendar_mark_day ()

```
void
gtk_calendar_mark_day (GtkCalendar *calendar,
                       guint day);
```

Places a visual marker on a particular day.

Parameters

calendar	a GtkCalendar
day	the day number to mark between 1 and 31.

---

gtk_calendar_unmark_day ()

```
void
gtk_calendar_unmark_day (GtkCalendar *calendar,
                         guint day);
```

Removes the visual marker from a particular day.

Parameters

calendar	a GtkCalendar.
day	the day number to unmark between 1 and 31.

---

gtk_calendar_get_day_is_marked ()

```
gboolean
gtk_calendar_get_day_is_marked (GtkCalendar *calendar,
                                guint day);
```

Returns if the `day` of the `calendar` is already marked.

Parameters

calendar	a GtkCalendar
day	the day number between 1 and 31.

Returns

whether the day is marked.

Since: 3.0

---

gtk_calendar_clear_marks ()

```
void
gtk_calendar_clear_marks (GtkCalendar *calendar);
```

Remove all visual markers.

Parameters

calendar	a GtkCalendar

---

gtk_calendar_get_display_options ()

```
GtkCalendarDisplayOptions
gtk_calendar_get_display_options (GtkCalendar *calendar);
```

Returns the current display options of `calendar`.

Parameters

calendar	a GtkCalendar

Returns

the display options.

Since: 2.4

---

gtk_calendar_set_display_options ()

```
void
gtk_calendar_set_display_options (GtkCalendar *calendar,
                                  GtkCalendarDisplayOptions flags);
```

Sets display options (whether to display the heading and the month headings).

Parameters

calendar	a GtkCalendar
flags	the display options to set

Since: 2.4

---

gtk_calendar_get_date ()

```
void
gtk_calendar_get_date (GtkCalendar *calendar,
```

```
                        guint *year,
                        guint *month,
                        guint *day);
```

Obtains the selected date from a GtkCalendar.

Parameters

calendar	a GtkCalendar	
year	location to store the year as a decimal number (e.g. 2011), or `NULL`.	[out][allow-none]
month	location to store the month number (between 0 and 11), or `NULL`.	[out][allow-none]
day	location to store the day number (between 1 and 31), or `NULL`.	[out][allow-none]

---

gtk_calendar_set_detail_func ()

```
void
gtk_calendar_set_detail_func (GtkCalendar *calendar,
                              GtkCalendarDetailFunc func,
                              gpointer data,
                              GDestroyNotify destroy);
```

Installs a function which provides Pango markup with detail information for each day. Examples for such details are holidays or appointments. That information is shown below each day when "show-details" is set. A tooltip containing with full detail information is provided, if the entire text should not fit into the details area, or if "show-details" is not set.

The size of the details area can be restricted by setting the "detail-width-chars" and "detail-height-rows" properties.

Parameters

calendar	a GtkCalendar.
func	a function providing details for each day.
data	data to pass to `func` invokations.
destroy	a function for releasing `data` .

Since: 2.14

---

gtk_calendar_get_detail_width_chars ()

```
gint
gtk_calendar_get_detail_width_chars (GtkCalendar *calendar);
```

Queries the width of detail cells, in characters. See "detail-width-chars".

Parameters

calendar	a GtkCalendar.

Returns

The width of detail cells, in characters.

Since: 2.14

---

gtk_calendar_set_detail_width_chars ()

```
void
gtk_calendar_set_detail_width_chars (GtkCalendar *calendar,
                                     gint chars);
```

Updates the width of detail cells. See “detail-width-chars”.

Parameters

calendar	a GtkCalendar.
chars	detail width in characters.

Since: 2.14

---

gtk_calendar_get_detail_height_rows ()

```
gint
gtk_calendar_get_detail_height_rows (GtkCalendar *calendar);
```

Queries the height of detail cells, in rows. See “detail-width-chars”.

Parameters

calendar	a GtkCalendar.

Returns

The height of detail cells, in rows.

Since: 2.14

---

gtk_calendar_set_detail_height_rows ()

```
void
gtk_calendar_set_detail_height_rows (GtkCalendar *calendar,
                                     gint rows);
```

Updates the height of detail cells. See “detail-height-rows”.

Parameters

calendar	a GtkCalendar.
rows	detail height in rows.

Since: 2.14

Types and Values

struct GtkCalendar

```
struct GtkCalendar;
```

---

enum GtkCalendarDisplayOptions

These options can be used to influence the display and behaviour of a GtkCalendar.

Members

GTK_CALENDAR_SHOW_HEADING	Specifies that the month and year should be displayed.
GTK_CALENDAR_SHOW_DAY_NAMES	Specifies that three letter day descriptions should be present.
GTK_CALENDAR_NO_MONTH_CHANGE	Prevents the user from switching months with the calendar.
GTK_CALENDAR_SHOW_WEEK_NUMBERS	Displays each week numbers of the current year, down the left side of the calendar.
GTK_CALENDAR_SHOW_DETAILS	Just show an indicator, not the full details text when details are provided. See `gtk_calendar_set_detail_func()`.

Property Details

The `"day"` property

`"day"` `gint`

The selected day (as a number between 1 and 31, or 0 to unselect the currently selected day). This property gets initially set to the current day.

Flags: Read / Write

Allowed values: [0,31]

Default value: 0

The “detail-height-rows” property

“detail-height-rows” gint

Height of a detail cell, in rows. A value of 0 allows any width. See gtk_calendar_set_detail_func().

Flags: Read / Write

Allowed values: [0,127]

Default value: 0

Since: 2.14

The “detail-width-chars” property

“detail-width-chars” gint

Width of a detail cell, in characters. A value of 0 allows any width. See gtk_calendar_set_detail_func().

Flags: Read / Write

Allowed values: [0,127]

Default value: 0

Since: 2.14

The “month” property

“month” gint

The selected month (as a number between 0 and 11). This property gets initially set to the current month.

Flags: Read / Write

Allowed values: [0,11]

Default value: 0

The “no-month-change” property

“no-month-change” gboolean

Determines whether the selected month can be changed.

Flags: Read / Write

Default value: FALSE

Since: 2.4

The “show-day-names” property

“show-day-names” gboolean

Determines whether day names are displayed.

Flags: Read / Write

Default value: TRUE

Since: 2.4

The “show-details” property

“show-details” gboolean

Determines whether details are shown directly in the widget, or if they are available only as tooltip. When this property is set days with details are marked.

Flags: Read / Write

Default value: TRUE

Since: 2.14

The “show-heading” property

“show-heading” gboolean

Determines whether a heading is displayed.

Flags: Read / Write

Default value: TRUE

Since: 2.4

The “show-week-numbers” property

“show-week-numbers” gboolean

Determines whether week numbers are displayed.

Flags: Read / Write

Default value: FALSE

Since: 2.4

The “year” property

“year” gint

The selected year. This property gets initially set to the current year.

Flags: Read / Write

Allowed values: [0,4194303]

Default value: 0

Style Property Details

The “horizontal-separation” style property

“horizontal-separation” gint

Separation between week headers and main area.

Flags: Read

Allowed values: >= 0

Default value: 4

The “inner-border” style property

“inner-border” gint

The spacing around the day/week headers and main area.

Flags: Read

Allowed values: >= 0

Default value: 4

The “vertical-separation” style property

“vertical-separation” gint

Space between day headers and main area.

Flags: Read

Allowed values: >= 0

Default value: 4

Signal Details

The “day-selected” signal

```
void
user_function (GtkCalendar *calendar,
               gpointer     user_data)
```

Emitted when the user selects a day.

Parameters

calendar	the object which received the signal.
user_data	user data set when the signal handler was connected.

Flags: Run First

The “day-selected-double-click” signal

```
void
user_function (GtkCalendar *calendar,
               gpointer     user_data)
```

Emitted when the user double-clicks a day.

Parameters

calendar	the object which received the signal.
user_data	user data set when the signal handler was connected.

Flags: Run First

The “month-changed” signal

```
void
user_function (GtkCalendar *calendar,
               gpointer     user_data)
```

Emitted when the user clicks a button to change the selected month on a calendar.

Parameters

calendar	the object which received the signal.
user_data	user data set when the signal handler was connected.

Flags: Run First

The “next-month” signal

```
void
```

```
user_function (GtkCalendar *calendar,
               gpointer     user_data)
```
Emitted when the user switched to the next month.

Parameters

calendar	the object which received the signal.
user_data	user data set when the signal handler was connected.

Flags: Run First

The `"next-year"` signal

```
void
user_function (GtkCalendar *calendar,
               gpointer     user_data)
```
Emitted when user switched to the next year.

Parameters

calendar	the object which received the signal.
user_data	user data set when the signal handler was connected.

Flags: Run First

The `"prev-month"` signal

```
void
user_function (GtkCalendar *calendar,
               gpointer     user_data)
```
Emitted when the user switched to the previous month.

Parameters

calendar	the object which received the signal.
user_data	user data set when the signal handler was connected.

Flags: Run First

The `"prev-year"` signal

```
void
user_function (GtkCalendar *calendar,
               gpointer     user_data)
```
Emitted when user switched to the previous year.

Parameters

calendar	the object which received the signal.
user_data	user data set when the signal handler was connected.

Flags: Run First

GtkDrawingArea

GtkDrawingArea — A widget for custom user interface elements

Functions

GtkWidget *	gtk_drawing_area_new ()

Types and Values

struct	GtkDrawingArea

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkDrawingArea
```

Implemented Interfaces

GtkDrawingArea implements AtkImplementorIface and GtkBuildable.

Includes

```
#include <gtk/gtk.h>
```

Description

The GtkDrawingArea widget is used for creating custom user interface elements. It's essentially a blank widget; you can draw on it. After creating a drawing area, the application may want to connect to:

- Mouse and button press signals to respond to input from the user. (Use `gtk_widget_add_events()` to enable events you wish to receive.)
- The "realize" signal to take any necessary actions when the widget is instantiated on a particular display. (Create GDK resources in response to this signal.)
- The "size-allocate" signal to take any necessary actions when the widget changes size.

- The “draw” signal to handle redrawing the contents of the widget.

The following code portion demonstrates using a drawing area to display a circle in the normal widget foreground color.

Note that GDK automatically clears the exposed area before sending the expose event, and that drawing is implicitly clipped to the exposed area. If you want to have a theme-provided background, you need to call `gtk_render_background()` in your ::draw method.

Simple GtkDrawingArea usage

```
gboolean
draw_callback (GtkWidget *widget, cairo_t
*cr, gpointer data)
{
  guint width, height;
  GdkRGBA color;
  GtkStyleContext *context;

  context = gtk_widget_get_style_context
(widget);

  width = gtk_widget_get_allocated_width
(widget);
  height = gtk_widget_get_allocated_height
(widget);

  gtk_render_background (context, cr, 0, 0,
width, height);

  cairo_arc (cr,
             width / 2.0, height / 2.0,
             MIN (width, height) / 2.0,
             0, 2 * G_PI);

  gtk_style_context_get_color (context,

gtk_style_context_get_state (context),
                                &color);
  gdk_cairo_set_source_rgba (cr, &color);

  cairo_fill (cr);

 return FALSE;
}
[...]
  GtkWidget *drawing_area =
gtk_drawing_area_new ();
  gtk_widget_set_size_request (drawing_area,
100, 100);
  g_signal_connect (G_OBJECT (drawing_area),
"draw",
                    G_CALLBACK
(draw_callback), NULL);
```

Draw signals are normally delivered when a drawing area first comes onscreen, or when it’s covered by another window and then uncovered. You can also force an expose event by adding to the “damage region” of the drawing area’s window; `gtk_widget_queue_draw_area()` and `gdk_window_invalidate_rect()` are equally good ways to do this. You’ll then get a draw signal for the invalid region.

The available routines for drawing are documented on the GDK Drawing Primitives page and the cairo

documentation.

To receive mouse events on a drawing area, you will need to enable them with `gtk_widget_add_events()`. To receive keyboard events, you will need to set the “can-focus” property on the drawing area, and you should probably draw some user-visible indication that the drawing area is focused. Use `gtk_widget_has_focus()` in your expose event handler to decide whether to draw the focus indicator. See `gtk_render_focus()` for one way to draw focus.

Functions

gtk_drawing_area_new ()

```
GtkWidget *
gtk_drawing_area_new (void);
```

Creates a new drawing area.

Returns

a new GtkDrawingArea

Types and Values

struct GtkDrawingArea

```
struct GtkDrawingArea;
```

See Also

GtkImage

GtkGLArea

GtkGLArea — A widget for custom drawing with OpenGL

Functions

GtkWidget *	gtk_gl_area_new ()
GdkGLContext *	gtk_gl_area_get_context ()
void	gtk_gl_area_make_current ()
void	gtk_gl_area_queue_render ()
void	gtk_gl_area_attach_buffers ()
void	gtk_gl_area_set_error ()
GError *	gtk_gl_area_get_error ()
void	gtk_gl_area_set_has_alpha ()
gboolean	gtk_gl_area_get_has_alpha ()
void	gtk_gl_area_set_has_depth_buffer ()
gboolean	gtk_gl_area_get_has_depth_buffer ()
void	gtk_gl_area_set_has_stencil_buffer ()
gboolean	gtk_gl_area_get_has_stencil_buffer ()
void	gtk_gl_area_set_auto_render ()
gboolean	gtk_gl_area_get_auto_render ()
void	gtk_gl_area_get_required_version ()
void	gtk_gl_area_set_required_version ()
void	gtk_gl_area_set_use_es ()
gboolean	gtk_gl_area_get_use_es ()

Properties

gboolean	auto-render	Read / Write
GdkGLContext *	context	Read
gboolean	has-alpha	Read / Write
gboolean	has-depth-buffer	Read / Write
gboolean	has-stencil-buffer	Read / Write
gboolean	use-es	Read / Write

Signals

GdkGLContext*	create-context	Run Last
gboolean	render	Run Last
void	resize	Run Last

Types and Values

struct	GtkGLArea
struct	GtkGLAreaClass

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkGLArea
```

Implemented Interfaces

GtkGLArea implements AtkImplementorIface and GtkBuildable.

Includes

```
#include <gtk/gtk.h>
```

Description

GtkGLArea is a widget that allows drawing with OpenGL.

GtkGLArea sets up its own GdkGLContext for the window it creates, and creates a custom GL framebuffer that the widget will do GL rendering onto. It also ensures that this framebuffer is the default GL rendering target when rendering.

In order to draw, you have to connect to the "render" signal, or subclass GtkGLArea and override the `GtkGLAreaClass.render()` virtual function.

The GtkGLArea widget ensures that the GdkGLContext is associated with the widget's drawing area, and it is kept updated when the size and position of the drawing area changes.

Drawing with GtkGLArea

The simplest way to draw using OpenGL commands in a GtkGLArea is to create a widget instance and connect to the "render" signal:

```
// create a GtkGLArea instance
GtkWidget *gl_area = gtk_gl_area_new ();

// connect to the "render" signal
g_signal_connect (gl_area, "render",
G_CALLBACK (render), NULL);
```

The `render()` function will be called when the GtkGLArea is ready for you to draw its content:

```
static gboolean
render (GtkGLArea *area, GdkGLContext
*context)
{
  // inside this function it's safe to use
GL; the given
  // #GdkGLContext has been made current to
the drawable
  // surface used by the #GtkGLArea and the
viewport has
  // already been set to be the size of the
allocation

  // we can start by clearing the buffer
  glClearColor (0, 0, 0, 0);
  glClear (GL_COLOR_BUFFER_BIT);

  // draw your object
  draw_an_object ();

  // we completed our drawing; the draw
commands will be
  // flushed at the end of the signal
```

```
  emission chain, and
    // the buffers will be drawn on the window
    return TRUE;
}
```

If you need to initialize OpenGL state, e.g. buffer objects or shaders, you should use the "realize" signal; you can use the "unrealize" signal to clean up. Since the GdkGLContext creation and initialization may fail, you will need to check for errors, using `gtk_gl_area_get_error()`. An example of how to safely initialize the GL state is:

```
static void
on_realize (GtkGLarea *area)
{
  // We need to make the context current if
we want to
  // call GL API
  gtk_gl_area_make_current (area);

  // If there were errors during the
initialization or
  // when trying to make the context current,
this
  // function will return a #GError for you
to catch
  if (gtk_gl_area_get_error (area) != NULL)
    return;

  // You can also use gtk_gl_area_set_error()
in order
  // to show eventual initialization errors
on the
  // GtkGLArea widget itself
  GError *internal_error = NULL;
  init_buffer_objects (&error);
  if (error != NULL)
    {
      gtk_gl_area_set_error (area, error);
      g_error_free (error);
      return;
    }

  init_shaders (&error);
  if (error != NULL)
    {
      gtk_gl_area_set_error (area, error);
      g_error_free (error);
      return;
    }
}
```

If you need to change the options for creating the GdkGLContext you should use the "create-context" signal.

Functions

gtk_gl_area_new ()

```
GtkWidget *
gtk_gl_area_new (void);
```

Creates a new GtkGLArea widget.

Returns

a new GtkGLArea

Since: 3.16

---

gtk_gl_area_get_context ()

```
GdkGLContext *
gtk_gl_area_get_context (GtkGLArea *area);
```

Retrieves the GdkGLContext used by `area` .

Parameters

area	a GtkGLArea

Returns

the GdkGLContext.

[transfer none]

Since: 3.16

---

gtk_gl_area_make_current ()

```
void
gtk_gl_area_make_current (GtkGLArea *area);
```

Ensures that the GdkGLContext used by `area` is associated with the GtkGLArea.

This function is automatically called before emitting the “render” signal, and doesn't normally need to be called by application code.

Parameters

area	a GtkGLArea

Since: 3.16

---

gtk_gl_area_queue_render ()

```
void
gtk_gl_area_queue_render (GtkGLArea *area);
```

Marks the currently rendered data (if any) as invalid, and queues a redraw of the widget, ensuring that the “render” signal is emitted during the draw.

This is only needed when the `gtk_gl_area_set_auto_render()` has been called with a `FALSE` value. The default behaviour is to emit “render” on each draw.

Parameters

area	a GtkGLArea	

Since: 3.16

---

gtk_gl_area_attach_buffers ()

```
void
gtk_gl_area_attach_buffers (GtkGLArea *area);
```

Ensures that the `area` framebuffer object is made the current draw and read target, and that all the required buffers for the `area` are created and bound to the frambuffer.

This function is automatically called before emitting the "render" signal, and doesn't normally need to be called by application code.

Parameters

area	a GtkGLArea	

Since: 3.16

---

gtk_gl_area_set_error ()

```
void
gtk_gl_area_set_error (GtkGLArea *area,
                       const GError *error);
```

Sets an error on the area which will be shown instead of the GL rendering. This is useful in the "create-context" signal if GL context creation fails.

Parameters

area	a GtkGLArea	
error	a new GError, or `NULL` to unset the error.	[allow-none]

Since: 3.16

---

gtk_gl_area_get_error ()

```
GError *
gtk_gl_area_get_error (GtkGLArea *area);
```

Gets the current error set on the `area` .

Parameters

area	a GtkGLArea

Returns

the GError or `NULL`.

[nullable][transfer none]

Since: 3.16

---

gtk_gl_area_set_has_alpha ()

```
void
gtk_gl_area_set_has_alpha (GtkGLArea *area,
                           gboolean has_alpha);
```

If `has_alpha` is `TRUE` the buffer allocated by the widget will have an alpha channel component, and when rendering to the window the result will be composited over whatever is below the widget.

If `has_alpha` is `FALSE` there will be no alpha channel, and the buffer will fully replace anything below the widget.

Parameters

area	a GtkGLArea
has_alpha	`TRUE` to add an alpha component

Since: 3.16

---

gtk_gl_area_get_has_alpha ()

```
gboolean
gtk_gl_area_get_has_alpha (GtkGLArea *area);
```

Returns whether the area has an alpha component.

Parameters

area	a GtkGLArea

Returns

`TRUE` if the `area` has an alpha component, `FALSE` otherwise

Since: 3.16

---

gtk_gl_area_set_has_depth_buffer ()

```
void
gtk_gl_area_set_has_depth_buffer (GtkGLArea *area,
                                  gboolean has_depth_buffer);
```

If `has_depth_buffer` is `TRUE` the widget will allocate and enable a depth buffer for the target framebuffer. Otherwise there will be none.

Parameters

area	a GtkGLArea
has_depth_buffer	`TRUE` to add a depth buffer

Since: 3.16

---

gtk_gl_area_get_has_depth_buffer ()

```
gboolean
gtk_gl_area_get_has_depth_buffer (GtkGLArea *area);
```

Returns whether the area has a depth buffer.

Parameters

area	a GtkGLArea

Returns

`TRUE` if the `area` has a depth buffer, `FALSE` otherwise

Since: 3.16

---

gtk_gl_area_set_has_stencil_buffer ()

```
void
gtk_gl_area_set_has_stencil_buffer (GtkGLArea *area,
                                    gboolean has_stencil_buffer);
```

If `has_stencil_buffer` is `TRUE` the widget will allocate and enable a stencil buffer for the target framebuffer. Otherwise there will be none.

Parameters

area	a GtkGLArea
has_stencil_buffer	`TRUE` to add a stencil buffer

Since: 3.16

---

gtk_gl_area_get_has_stencil_buffer ()

```
gboolean
gtk_gl_area_get_has_stencil_buffer (GtkGLArea *area);
```

Returns whether the area has a stencil buffer.

Parameters

area	a GtkGLArea

Returns

`TRUE` if the `area` has a stencil buffer, `FALSE` otherwise

Since: 3.16

---

gtk_gl_area_set_auto_render ()

```
void
gtk_gl_area_set_auto_render (GtkGLArea *area,
                             gboolean auto_render);
```

If `auto_render` is `TRUE` the "render" signal will be emitted every time the widget draws. This is the default and is useful if drawing the widget is faster.

If `auto_render` is `FALSE` the data from previous rendering is kept around and will be used for drawing the widget the next time, unless the window is resized. In order to force a rendering `gtk_gl_area_queue_render()` must be called. This mode is useful when the scene changes seldomly, but takes a long time to redraw.

Parameters

area	a GtkGLArea
auto_render	a boolean

Since: 3.16

---

gtk_gl_area_get_auto_render ()

```
gboolean
gtk_gl_area_get_auto_render (GtkGLArea *area);
```

Returns whether the area is in auto render mode or not.

Parameters

area	a GtkGLArea

Returns

`TRUE` if the `area` is auto rendering, `FALSE` otherwise

Since: 3.16

---

gtk_gl_area_get_required_version ()

```
void
gtk_gl_area_get_required_version (GtkGLArea *area,
                                  gint *major,
                                  gint *minor);
```

Retrieves the required version of OpenGL set using `gtk_gl_area_set_required_version()`.

Parameters

area	a GtkGLArea	
major	return location for the required major version.	[out]
minor	return location for the required minor version.	[out]

Since: 3.16

---

gtk_gl_area_set_required_version ()

```
void
gtk_gl_area_set_required_version (GtkGLArea *area,
                                  gint major,
                                  gint minor);
```

Sets the required version of OpenGL to be used when creating the context for the widget.

This function must be called before the area has been realized.

Parameters

area	a GtkGLArea
major	the major version
minor	the minor version

Since: 3.16

---

gtk_gl_area_set_use_es ()

```
void
gtk_gl_area_set_use_es (GtkGLArea *area,
                        gboolean use_es);
```

Sets whether the `area` should create an OpenGL or an OpenGL ES context.

You should check the capabilities of the GdkGLContext before drawing with either API.

Parameters

area	a GtkGLArea
use_es	whether to use OpenGL or OpenGL ES

Since: 3.22

---

gtk_gl_area_get_use_es ()

```
gboolean
gtk_gl_area_get_use_es (GtkGLArea *area);
```

Retrieves the value set by `gtk_gl_area_set_use_es()`.

Parameters

area	a GtkGLArea

Returns

`TRUE` if the GtkGLArea should create an OpenGL ES context and `FALSE` otherwise

Since: 3.22

Types and Values

struct GtkGLArea

```
struct GtkGLArea;
```

A GtkWidget used for drawing with OpenGL.

Since: 3.16

---

struct GtkGLAreaClass

```
struct GtkGLAreaClass {
  gboolean       (* render)         (GtkGLArea        *area,
                                     GdkGLContext     *context);
  void           (* resize)         (GtkGLArea        *area,
                                     int               width,
                                     int               height);
  GdkGLContext * (* create_context) (GtkGLArea        *area);
};
```

The `GtkGLAreaClass` structure contains only private data.

Members

`render` ()	class closure for the “render” signal
`resize` ()	class closeure for the “resize” signal
`create_context` ()	class closure for the “create-context” signal

Since: 3.16

Property Details

The “auto-render” property

`“auto-render”` `gboolean`

If set to `TRUE` the “render” signal will be emitted every time the widget draws. This is the default and is useful if drawing the widget is faster.

If set to `FALSE` the data from previous rendering is kept around and will be used for drawing the widget the next time, unless the window is resized. In order to force a rendering `gtk_gl_area_queue_render()` must be called. This mode is useful when the scene changes seldomly, but takes a long time to redraw.

Flags: Read / Write

Default value: TRUE

Since: 3.16

The “context” property

`“context”` `GdkGLContext *`

The GdkGLContext used by the GtkGLArea widget.

The GtkGLArea widget is responsible for creating the GdkGLContext instance. If you need to render with other kinds of buffers (stencil, depth, etc), use render buffers.

Flags: Read

Since: 3.16

The “has-alpha” property

`“has-alpha”` `gboolean`

If set to `TRUE` the buffer allocated by the widget will have an alpha channel component, and when rendering to the window the result will be composited over whatever is below the widget.

If set to `FALSE` there will be no alpha channel, and the buffer will fully replace anything below the widget.

Flags: Read / Write

Default value: FALSE

Since: 3.16

The “has-depth-buffer” property

“has-depth-buffer” gboolean

If set to TRUE the widget will allocate and enable a depth buffer for the target framebuffer.

Flags: Read / Write

Default value: FALSE

Since: 3.16

The “has-stencil-buffer” property

“has-stencil-buffer” gboolean

If set to TRUE the widget will allocate and enable a stencil buffer for the target framebuffer.

Flags: Read / Write

Default value: FALSE

Since: 3.16

The “use-es” property

“use-es” gboolean

If set to TRUE the widget will try to create a GdkGLContext using OpenGL ES instead of OpenGL.

See also: gdk_gl_context_set_use_es()

Flags: Read / Write

Default value: FALSE

Since: 3.22

Signal Details

The “create-context” signal

```
GdkGLContext*
user_function (GtkGLArea *area,
               gpointer   user_data)
```

The ::create-context signal is emitted when the widget is being realized, and allows you to override how the GL context is created. This is useful when you want to reuse an existing GL context, or if you want to try creating

different kinds of GL options.

If context creation fails then the signal handler can use `gtk_gl_area_set_error()` to register a more detailed error of how the construction failed.

Parameters

area	the GtkGLArea that emitted the signal	
error	location to store error information on failure.	[allow-none]
user_data	user data set when the signal handler was connected.	

Returns

a newly created GdkGLContext; the GtkGLArea widget will take ownership of the returned value.

[transfer full]

Flags: Run Last

Since: 3.16

The `"render"` signal

```
gboolean
user_function (GtkGLArea    *area,
               GdkGLContext *context,
               gpointer      user_data)
```

The ::render signal is emitted every time the contents of the GtkGLArea should be redrawn.

The `context` is bound to the `area` prior to emitting this function, and the buffers are painted to the window once the emission terminates.

Parameters

area	the GtkGLArea that emitted the signal
context	the GdkGLContext used by `area`
user_data	user data set when the signal handler was connected.

Returns

`TRUE` to stop other handlers from being invoked for the event. `FALSE` to propagate the event further.

Flags: Run Last

Since: 3.16

The "resize" signal

```
void
user_function (GtkGLArea *area,
               gint       width,
               gint       height,
               gpointer   user_data)
```

The ::resize signal is emitted once when the widget is realized, and then each time the widget is changed while realized. This is useful in order to keep GL state up to date with the widget size, like for instance camera properties which may depend on the width/height ratio.

The GL context for the area is guaranteed to be current when this signal is emitted.

The default handler sets up the GL viewport.

Parameters

area	the GtkGLArea that emitted the signal
width	the width of the viewport
height	the height of the viewport
user_data	user data set when the signal handler was connected.

Flags: Run Last

Since: 3.16

GtkEventBox

GtkEventBox — A widget used to catch events for widgets which do not have their own window

Functions

GtkWidget *	gtk_event_box_new ()
void	gtk_event_box_set_above_child ()
gboolean	gtk_event_box_get_above_child ()
void	gtk_event_box_set_visible_window ()
gboolean	gtk_event_box_get_visible_window ()

Properties

gboolean	above-child	Read / Write
gboolean	visible-window	Read / Write

Types and Values

struct	GtkEventBox

struct GtkEventBoxClass

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkContainer
            └── GtkBin
                └── GtkEventBox
```

Implemented Interfaces

GtkEventBox implements AtkImplementorIface and GtkBuildable.

Includes

```
#include <gtk/gtk.h>
```

Description

The GtkEventBox widget is a subclass of GtkBin which also has its own window. It is useful since it allows you to catch events for widgets which do not have their own window.

Functions

gtk_event_box_new ()

```
GtkWidget *
gtk_event_box_new (void);
```

Creates a new GtkEventBox.

Returns

a new GtkEventBox

---

gtk_event_box_set_above_child ()

```
void
gtk_event_box_set_above_child (GtkEventBox *event_box,
                               gboolean above_child);
```

Set whether the event box window is positioned above the windows of its child, as opposed to below it. If the window is above, all events inside the event box will go to the event box. If the window is below, events in windows of child widgets will first got to that widget, and then to its parents.

The default is to keep the window below the child.

Parameters

event_box	a GtkEventBox
above_child	TRUE if the event box window is above its child

Since: 2.4

---

gtk_event_box_get_above_child ()

```
gboolean
gtk_event_box_get_above_child (GtkEventBox *event_box);
```

Returns whether the event box window is above or below the windows of its child. See gtk_event_box_set_above_child() for details.

Parameters

event_box	a GtkEventBox

Returns

TRUE if the event box window is above the window of its child

Since: 2.4

---

gtk_event_box_set_visible_window ()

```
void
gtk_event_box_set_visible_window (GtkEventBox *event_box,
                                  gboolean visible_window);
```

Set whether the event box uses a visible or invisible child window. The default is to use visible windows.

In an invisible window event box, the window that the event box creates is a GDK_INPUT_ONLY window, which means that it is invisible and only serves to receive events.

A visible window event box creates a visible (GDK_INPUT_OUTPUT) window that acts as the parent window for all the widgets contained in the event box.

You should generally make your event box invisible if you just want to trap events. Creating a visible window may cause artifacts that are visible to the user, especially if the user is using a theme with gradients or pixmaps.

The main reason to create a non input-only event box is if you want to set the background to a different color or draw on it.

There is one unexpected issue for an invisible event box that has its window below the child. (See gtk_event_box_set_above_child().) Since the input-only window is not an ancestor window of any windows that descendent widgets of the event box create, events on these windows aren't propagated up by the windowing system, but only by GTK+. The practical effect of this is if an event isn't in the event mask for the descendant window (see gtk_widget_add_events()), it won't be received by the event box.

This problem doesn’t occur for visible event boxes, because in that case, the event box window is actually the ancestor of the descendant windows, not just at the same place on the screen.

Parameters

event_box	a GtkEventBox
visible_window	TRUE to make the event box have a visible window

Since: 2.4

---

gtk_event_box_get_visible_window ()

```
gboolean
gtk_event_box_get_visible_window (GtkEventBox *event_box);
```

Returns whether the event box has a visible window. See gtk_event_box_set_visible_window() for details.

Parameters

event_box	a GtkEventBox

Returns

TRUE if the event box window is visible

Since: 2.4

Types and Values

struct GtkEventBox

```
struct GtkEventBox;
```

---

struct GtkEventBoxClass

```
struct GtkEventBoxClass {
  GtkBinClass parent_class;
};
```

Members

Property Details

The "above-child" property

"above-child" gboolean

Whether the event-trapping window of the eventbox is above the window of the child widget as opposed to below it.

Flags: Read / Write

Default value: FALSE

The "visible-window" property

"visible-window" gboolean

Whether the event box is visible, as opposed to invisible and only used to trap events.

Flags: Read / Write

Default value: TRUE

GtkHandleBox

GtkHandleBox — a widget for detachable window portions

Functions

GtkWidget *	gtk_handle_box_new ()
void	gtk_handle_box_set_shadow_type ()
void	gtk_handle_box_set_handle_position ()
void	gtk_handle_box_set_snap_edge ()
GtkPositionType	gtk_handle_box_get_handle_position ()
GtkShadowType	gtk_handle_box_get_shadow_type ()
GtkPositionType	gtk_handle_box_get_snap_edge ()
gboolean	gtk_handle_box_get_child_detached ()

Properties

gboolean	child-detached	Read
GtkPositionType	handle-position	Read / Write
GtkShadowType	shadow-type	Read / Write
GtkPositionType	snap-edge	Read / Write
gboolean	snap-edge-set	Read / Write

Signals

void	child-attached	Run First
void	child-detached	Run First

Types and Values

struct	GtkHandleBox
struct	GtkHandleBoxClass

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkContainer
            └── GtkBin
                └── GtkHandleBox
```

Implemented Interfaces

GtkHandleBox implements AtkImplementorIface and GtkBuildable.

Includes

```
#include <gtk/gtk.h>
```

Description

The GtkHandleBox widget allows a portion of a window to be "torn off". It is a bin widget which displays its child and a handle that the user can drag to tear off a separate window (the “float window”) containing the child widget. A thin “ghost” is drawn in the original location of the handlebox. By dragging the separate window back to its original location, it can be reattached.

When reattaching, the ghost and float window, must be aligned along one of the edges, the “snap edge”. This either can be specified by the application programmer explicitly, or GTK+ will pick a reasonable default based on the handle position.

To make detaching and reattaching the handlebox as minimally confusing as possible to the user, it is important to set the snap edge so that the snap edge does not move when the handlebox is deattached. For instance, if the handlebox is packed at the bottom of a VBox, then when the handlebox is detached, the bottom edge of the handlebox's allocation will remain fixed as the height of the handlebox shrinks, so the snap edge should be set to `GTK_POS_BOTTOM`.

> GtkHandleBox has been deprecated. It is very specialized, lacks features to make it useful and most importantly does not fit well into modern application design. Do not use it. There is no replacement.

Functions

gtk_handle_box_new ()

```
GtkWidget *
gtk_handle_box_new (void);
```

`gtk_handle_box_new` has been deprecated since version 3.4 and should not be used in newly-written code.

GtkHandleBox has been deprecated.

Create a new handle box.

Returns

a new GtkHandleBox.

---

gtk_handle_box_set_shadow_type ()

```
void
gtk_handle_box_set_shadow_type (GtkHandleBox *handle_box,
                                GtkShadowType type);
```

`gtk_handle_box_set_shadow_type` has been deprecated since version 3.4 and should not be used in newly-written code.

GtkHandleBox has been deprecated.

Sets the type of shadow to be drawn around the border of the handle box.

Parameters

handle_box	a GtkHandleBox
type	the shadow type.

---

gtk_handle_box_set_handle_position ()

```
void
gtk_handle_box_set_handle_position (GtkHandleBox *handle_box,
                                    GtkPositionType position);
```

`gtk_handle_box_set_handle_position` has been deprecated since version 3.4 and should not be used in newly-written code.

GtkHandleBox has been deprecated.

Sets the side of the handlebox where the handle is drawn.

Parameters

handle_box	a GtkHandleBox
position	the side of the handlebox where the handle should be drawn.

---

gtk_handle_box_set_snap_edge ()

```
void
gtk_handle_box_set_snap_edge (GtkHandleBox *handle_box,
                              GtkPositionType edge);
```

`gtk_handle_box_set_snap_edge` has been deprecated since version 3.4 and should not be used in newly-written code.

GtkHandleBox has been deprecated.

Sets the snap edge of a handlebox. The snap edge is the edge of the detached child that must be aligned with the corresponding edge of the “ghost” left behind when the child was detached to reattach the torn-off window. Usually, the snap edge should be chosen so that it stays in the same place on the screen when the handlebox is torn off.

If the snap edge is not set, then an appropriate value will be guessed from the handle position. If the handle position is `GTK_POS_RIGHT` or `GTK_POS_LEFT`, then the snap edge will be `GTK_POS_TOP`, otherwise it will be `GTK_POS_LEFT`.

Parameters

handle_box	a GtkHandleBox
edge	the snap edge, or -1 to unset the value; in which case GTK+ will try to guess an appropriate value in the future.

---

gtk_handle_box_get_handle_position ()

```
GtkPositionType
gtk_handle_box_get_handle_position (GtkHandleBox *handle_box);
```

`gtk_handle_box_get_handle_position` has been deprecated since version 3.4 and should not be used in newly-written code.

GtkHandleBox has been deprecated.

Gets the handle position of the handle box. See `gtk_handle_box_set_handle_position()`.

Parameters

handle_box	a GtkHandleBox

Returns

the current handle position.

---

gtk_handle_box_get_shadow_type ()

```
GtkShadowType
gtk_handle_box_get_shadow_type (GtkHandleBox *handle_box);
```

`gtk_handle_box_get_shadow_type` has been deprecated since version 3.4 and should not be used in newly-written code.

GtkHandleBox has been deprecated.

Gets the type of shadow drawn around the handle box. See `gtk_handle_box_set_shadow_type()`.

Parameters

handle_box	a GtkHandleBox

Returns

the type of shadow currently drawn around the handle box.

---

gtk_handle_box_get_snap_edge ()

```
GtkPositionType
gtk_handle_box_get_snap_edge (GtkHandleBox *handle_box);
```

`gtk_handle_box_get_snap_edge` has been deprecated since version 3.4 and should not be used in newly-written code.

GtkHandleBox has been deprecated.

Gets the edge used for determining reattachment of the handle box. See `gtk_handle_box_set_snap_edge()`.

Parameters

handle_box	a GtkHandleBox

Returns

the edge used for determining reattachment, or (GtkPositionType)-1 if this is determined (as per default) from the handle position.

---

gtk_handle_box_get_child_detached ()

```
gboolean
gtk_handle_box_get_child_detached (GtkHandleBox *handle_box);
```

`gtk_handle_box_get_child_detached` has been deprecated since version 3.4 and should not be used in newly-written code.

GtkHandleBox has been deprecated.

Whether the handlebox's child is currently detached.

Parameters

handle_box	a GtkHandleBox

Returns

TRUE if the child is currently detached, otherwise FALSE

Since: 2.14

Types and Values

struct GtkHandleBox

```
struct GtkHandleBox;
```

---

struct GtkHandleBoxClass

```
struct GtkHandleBoxClass {
  GtkBinClass parent_class;

  void (*child_attached) (GtkHandleBox *handle_box,
                          GtkWidget *child);
  void (*child_detached) (GtkHandleBox *handle_box,
                          GtkWidget *child);
};
```

Members

child_attached ()	Signal emitted when the contents of the handlebox are reattached to the main window. Deprecated: 3.4.
child_detached ()	Signal emitted when the contents of the handlebox are detached from the main window. Deprecated: 3.4.

Property Details

The “child-detached” property

```
  “child-detached”           gboolean
```

A boolean value indicating whether the handlebox's child is attached or detached.

Flags: Read

Default value: FALSE

The “handle-position” property

“handle-position” GtkPositionType

Position of the handle relative to the child widget.

Flags: Read / Write

Default value: GTK_POS_LEFT

The “shadow-type” property

“shadow-type” GtkShadowType

Appearance of the shadow that surrounds the container.

Flags: Read / Write

Default value: GTK_SHADOW_OUT

The “snap-edge” property

“snap-edge” GtkPositionType

Side of the handlebox that's lined up with the docking point to dock the handlebox.

Flags: Read / Write

Default value: GTK_POS_TOP

The “snap-edge-set” property

“snap-edge-set” gboolean

Whether to use the value from the snap_edge property or a value derived from handle_position.

Flags: Read / Write

Default value: FALSE

Signal Details

The “child-attached” signal

```
void
user_function (GtkHandleBox *handlebox,
               GtkWidget    *widget,
               gpointer      user_data)
```

This signal is emitted when the contents of the handlebox are reattached to the main window.

`GtkHandleBox::child-attached` has been deprecated since version 3.4 and should not be used in newly-written code.

GtkHandleBox has been deprecated.

Parameters

handlebox	the object which received the signal.
widget	the child widget of the handlebox. (this argument provides no extra information and is here only for backwards-compatibility)
user_data	user data set when the signal handler was connected.

Flags: Run First

The `"child-detached"` signal

```
void
user_function (GtkHandleBox *handlebox,
               GtkWidget    *widget,
               gpointer      user_data)
```

This signal is emitted when the contents of the handlebox are detached from the main window.

`GtkHandleBox::child-detached` has been deprecated since version 3.4 and should not be used in newly-written code.

GtkHandleBox has been deprecated.

Parameters

handlebox	the object which received the signal.
widget	the child widget of the handlebox. (this argument provides no extra information and is here only for backwards-compatibility)
user_data	user data set when the signal handler was connected.

Flags: Run First

GtkIMContextSimple

GtkIMContextSimple — An input method context supporting table-based input methods

Functions

GtkIMContext *	gtk_im_context_simple_new ()
void	gtk_im_context_simple_add_table ()

Types and Values

struct	GtkIMContextSimple
#define	GTK_MAX_COMPOSE_LEN

Object Hierarchy

```
GObject
└── GtkIMContext
    └── GtkIMContextSimple
```

Includes

```
#include <gtk/gtk.h>
```

Description

GtkIMContextSimple is a simple input method context supporting table-based input methods. It has a built-in table of compose sequences that is derived from the X11 Compose files.

GtkIMContextSimple reads additional compose sequences from the first of the following files that is found: ~/.config/gtk-3.0/Compose, ~/.XCompose, /usr/share/X11/locale/$locale/Compose (for locales that have a nontrivial Compose file). The syntax of these files is described in the Compose(5) manual page.

GtkIMContextSimple also supports numeric entry of Unicode characters by typing Ctrl-Shift-u, followed by a hexadecimal Unicode codepoint. For example, Ctrl-Shift-u 1 2 3 Enter yields U+0123 LATIN SMALL LETTER G WITH CEDILLA, i.e. ģ.

Functions

gtk_im_context_simple_new ()

```
GtkIMContext *
gtk_im_context_simple_new (void);
```

Creates a new GtkIMContextSimple.

Returns

a new GtkIMContextSimple.

gtk_im_context_simple_add_table ()

```
void
gtk_im_context_simple_add_table (GtkIMContextSimple *context_simple,
                                 guint16 *data,
                                 gint max_seq_len,
                                 gint n_seqs);
```

Adds an additional table to search to the input context. Each row of the table consists of `max_seq_len` key symbols followed by two guint16 interpreted as the high and low words of a gunicode value. Tables are searched starting from the last added.

The table must be sorted in dictionary order on the numeric value of the key symbol fields. (Values beyond the length of the sequence should be zero.)

[skip]

Parameters

context_simple	A GtkIMContextSimple	
data	the table.	[array]
max_seq_len	Maximum length of a sequence in the table (cannot be greater than GTK_MAX_COMPOSE_LEN)	
n_seqs	number of sequences in the table	

Types and Values

struct GtkIMContextSimple

```
struct GtkIMContextSimple;
```

GTK_MAX_COMPOSE_LEN

```
#define GTK_MAX_COMPOSE_LEN 7
```

The maximum length of sequences in compose tables.

GtkIMMulticontext

GtkIMMulticontext — An input method context supporting multiple, loadable input methods

Functions

GtkIMContext *	gtk_im_multicontext_new ()
void	gtk_im_multicontext_append_menuitems ()
const char *	gtk_im_multicontext_get_context_id ()
void	gtk_im_multicontext_set_context_id ()

Types and Values

struct	GtkIMMulticontext

Object Hierarchy

```
GObject
└── GtkIMContext
    └── GtkIMMulticontext
```

Includes

```
#include <gtk/gtk.h>
```

Description

Functions

gtk_im_multicontext_new ()

```
GtkIMContext *
gtk_im_multicontext_new (void);
```

Creates a new GtkIMMulticontext.

Returns

a new GtkIMMulticontext.

---

gtk_im_multicontext_append_menuitems ()

```
void
gtk_im_multicontext_append_menuitems (GtkIMMulticontext *context,
                                      GtkMenuShell *menushell);
```

`gtk_im_multicontext_append_menuitems` has been deprecated since version 3.10 and should not be used in newly-written code.

It is better to use the system-wide input method framework for changing input methods. Modern desktop shells offer on-screen displays for this that can triggered with a keyboard shortcut, e.g. Super-Space.

Add menuitems for various available input methods to a menu; the menuitems, when selected, will switch the input method for the context and the global default input method.

Parameters

context	a GtkIMMulticontext
menushell	a GtkMenuShell

---

gtk_im_multicontext_get_context_id ()

```
const char *
gtk_im_multicontext_get_context_id (GtkIMMulticontext *context);
```

Gets the id of the currently active slave of the `context` .

Parameters

context	a GtkIMMulticontext

Returns

the id of the currently active slave

Since: 2.16

---

gtk_im_multicontext_set_context_id ()

```
void
gtk_im_multicontext_set_context_id (GtkIMMulticontext *context,
                                    const char *context_id);
```

Sets the context id for `context` .

This causes the currently active slave of `context` to be replaced by the slave corresponding to the new context id.

Parameters

context	a GtkIMMulticontext
context_id	the id to use

Since: 2.16

Types and Values

struct GtkIMMulticontext

```
struct GtkIMMulticontext;
```

GtkSizeGroup

GtkSizeGroup — Grouping widgets so they request the same size

Functions

GtkSizeGroup *	gtk_size_group_new ()
void	gtk_size_group_set_mode ()
GtkSizeGroupMode	gtk_size_group_get_mode ()
void	gtk_size_group_set_ignore_hidden ()
gboolean	gtk_size_group_get_ignore_hidden ()
void	gtk_size_group_add_widget ()
void	gtk_size_group_remove_widget ()
GSList *	gtk_size_group_get_widgets ()

Properties

gboolean	ignore-hidden	Read / Write
GtkSizeGroupMode	mode	Read / Write

Types and Values

struct	GtkSizeGroup
enum	GtkSizeGroupMode

Object Hierarchy

```
GObject
└── GtkSizeGroup
```

Implemented Interfaces

GtkSizeGroup implements GtkBuildable.

Includes

```
#include <gtk/gtk.h>
```

Description

GtkSizeGroup provides a mechanism for grouping a number of widgets together so they all request the same amount of space. This is typically useful when you want a column of widgets to have the same size, but you can't use a GtkGrid widget.

In detail, the size requested for each widget in a GtkSizeGroup is the maximum of the sizes that would have been requested for each widget in the size group if they were not in the size group. The mode of the size group (see `gtk_size_group_set_mode()`) determines whether this applies to the horizontal size, the vertical size, or

both sizes.

Note that size groups only affect the amount of space requested, not the size that the widgets finally receive. If you want the widgets in a GtkSizeGroup to actually be the same size, you need to pack them in such a way that they get the size they request and not more. For example, if you are packing your widgets into a table, you would not include the GTK_FILL flag.

GtkSizeGroup objects are referenced by each widget in the size group, so once you have added all widgets to a GtkSizeGroup, you can drop the initial reference to the size group with g_object_unref(). If the widgets in the size group are subsequently destroyed, then they will be removed from the size group and drop their references on the size group; when all widgets have been removed, the size group will be freed.

Widgets can be part of multiple size groups; GTK+ will compute the horizontal size of a widget from the horizontal requisition of all widgets that can be reached from the widget by a chain of size groups of type GTK_SIZE_GROUP_HORIZONTAL or GTK_SIZE_GROUP_BOTH, and the vertical size from the vertical requisition of all widgets that can be reached from the widget by a chain of size groups of type GTK_SIZE_GROUP_VERTICAL or GTK_SIZE_GROUP_BOTH.

Note that only non-contextual sizes of every widget are ever consulted by size groups (since size groups have no knowledge of what size a widget will be allocated in one dimension, it cannot derive how much height a widget will receive for a given width). When grouping widgets that trade height for width in mode GTK_SIZE_GROUP_VERTICAL or GTK_SIZE_GROUP_BOTH: the height for the minimum width will be the requested height for all widgets in the group. The same is of course true when horizontally grouping width for height widgets.

Widgets that trade height-for-width should set a reasonably large minimum width by way of “width-chars” for instance. Widgets with static sizes as well as widgets that grow (such as ellipsizing text) need no such considerations.

GtkSizeGroup as GtkBuildable

Size groups can be specified in a UI definition by placing an <object> element with class="GtkSizeGroup" somewhere in the UI definition. The widgets that belong to the size group are specified by a <widgets> element that may contain multiple <widget> elements, one for each member of the size group. The ”name” attribute gives the id of the widget.

An example of a UI definition fragment with GtkSizeGroup:

```
<object class="GtkSizeGroup">
  <property
name="mode">GTK_SIZE_GROUP_HORIZONTAL</proper
ty>
  <widgets>
    <widget name="radio1"/>
    <widget name="radio2"/>
  </widgets>
</object>
```

Functions

gtk_size_group_new ()

```
GtkSizeGroup *
```

```
gtk_size_group_new (GtkSizeGroupMode mode);
```
Create a new GtkSizeGroup.

Parameters

mode	the mode for the new size group.

Returns

a newly created GtkSizeGroup

---

gtk_size_group_set_mode ()

```
void
gtk_size_group_set_mode (GtkSizeGroup *size_group,
                         GtkSizeGroupMode mode);
```
Sets the GtkSizeGroupMode of the size group. The mode of the size group determines whether the widgets in the size group should all have the same horizontal requisition (GTK_SIZE_GROUP_HORIZONTAL) all have the same vertical requisition (GTK_SIZE_GROUP_VERTICAL), or should all have the same requisition in both directions (GTK_SIZE_GROUP_BOTH).

Parameters

size_group	a GtkSizeGroup
mode	the mode to set for the size group.

---

gtk_size_group_get_mode ()

```
GtkSizeGroupMode
gtk_size_group_get_mode (GtkSizeGroup *size_group);
```
Gets the current mode of the size group. See gtk_size_group_set_mode().

Parameters

size_group	a GtkSizeGroup

Returns

the current mode of the size group.

---

gtk_size_group_set_ignore_hidden ()

```
void
```

```
gtk_size_group_set_ignore_hidden (GtkSizeGroup *size_group,
                                  gboolean ignore_hidden);
```

`gtk_size_group_set_ignore_hidden` has been deprecated since version 3.22 and should not be used in newly-written code.

Measuring the size of hidden widgets has not worked reliably for a long time. In most cases, they will report a size of 0 nowadays, and thus, their size will not affect the other size group members. In effect, size groups will always operate as if this property was `TRUE`. Use a GtkStack instead to hide widgets while still having their size taken into account.

Sets whether unmapped widgets should be ignored when calculating the size.

Parameters

size_group	a GtkSizeGroup
ignore_hidden	whether unmapped widgets should be ignored when calculating the size

Since: 2.8

---

gtk_size_group_get_ignore_hidden ()

```
gboolean
gtk_size_group_get_ignore_hidden (GtkSizeGroup *size_group);
```

`gtk_size_group_get_ignore_hidden` has been deprecated since version 3.22 and should not be used in newly-written code.

Measuring the size of hidden widgets has not worked reliably for a long time. In most cases, they will report a size of 0 nowadays, and thus, their size will not affect the other size group members. In effect, size groups will always operate as if this property was `TRUE`. Use a GtkStack instead to hide widgets while still having their size taken into account.

Returns if invisible widgets are ignored when calculating the size.

Parameters

size_group	a GtkSizeGroup

Returns

`TRUE` if invisible widgets are ignored.

Since: 2.8

---

gtk_size_group_add_widget ()

```
void
gtk_size_group_add_widget (GtkSizeGroup *size_group,
                           GtkWidget *widget);
```

Adds a widget to a GtkSizeGroup. In the future, the requisition of the widget will be determined as the maximum of its requisition and the requisition of the other widgets in the size group. Whether this applies horizontally, vertically, or in both directions depends on the mode of the size group. See `gtk_size_group_set_mode()`.

When the widget is destroyed or no longer referenced elsewhere, it will be removed from the size group.

Parameters

size_group	a GtkSizeGroup
widget	the GtkWidget to add

---

gtk_size_group_remove_widget ()

```
void
gtk_size_group_remove_widget (GtkSizeGroup *size_group,
                              GtkWidget *widget);
```

Removes a widget from a GtkSizeGroup.

Parameters

size_group	a GtkSizeGroup
widget	the GtkWidget to remove

---

gtk_size_group_get_widgets ()

```
GSList *
gtk_size_group_get_widgets (GtkSizeGroup *size_group);
```

Returns the list of widgets associated with `size_group` .

Parameters

size_group	a GtkSizeGroup

Returns

a GSList of widgets. The list is owned by GTK+ and should not be modified.

[element-type GtkWidget][transfer none]

Since: 2.10

Types and Values

struct GtkSizeGroup

```
struct GtkSizeGroup;
```

---

enum GtkSizeGroupMode

The mode of the size group determines the directions in which the size group affects the requested sizes of its component widgets.

Members

GTK_SIZE_GROUP_NONE	group has no effect
GTK_SIZE_GROUP_HORIZONTAL	group affects horizontal requisition
GTK_SIZE_GROUP_VERTICAL	group affects vertical requisition
GTK_SIZE_GROUP_BOTH	group affects both horizontal and vertical requisition

Property Details

The `"ignore-hidden"` property

`"ignore-hidden"` `gboolean`

If `TRUE`, unmapped widgets are ignored when determining the size of the group.

`GtkSizeGroup:ignore-hidden` has been deprecated since version 3.22 and should not be used in newly-written code.

Measuring the size of hidden widgets has not worked reliably for a long time. In most cases, they will report a size of 0 nowadays, and thus, their size will not affect the other size group members. In effect, size groups will always operate as if this property was `TRUE`. Use a GtkStack instead to hide widgets while still having their size taken into account.

Flags: Read / Write

Default value: FALSE

Since: 2.8

The `"mode"` property

`"mode"` `GtkSizeGroupMode`

The directions in which the size group affects the requested sizes of its component widgets.

Flags: Read / Write

Default value: GTK_SIZE_GROUP_HORIZONTAL

GtkTooltip

GtkTooltip — Add tips to your widgets

Functions

void	gtk_tooltip_set_markup ()
void	gtk_tooltip_set_text ()
void	gtk_tooltip_set_icon ()
void	gtk_tooltip_set_icon_from_stock ()
void	gtk_tooltip_set_icon_from_icon_name ()
void	gtk_tooltip_set_icon_from_gicon ()
void	gtk_tooltip_set_custom ()
void	gtk_tooltip_trigger_tooltip_query ()
void	gtk_tooltip_set_tip_area ()

Types and Values

GtkTooltip

Object Hierarchy

```
GObject
└── GtkTooltip
```

Includes

```
#include <gtk/gtk.h>
```

Description

Basic tooltips can be realized simply by using `gtk_widget_set_tooltip_text()` or `gtk_widget_set_tooltip_markup()` without any explicit tooltip object.

When you need a tooltip with a little more fancy contents, like adding an image, or you want the tooltip to have different contents per GtkTreeView row or cell, you will have to do a little more work:

- Set the "has-tooltip" property to `TRUE`, this will make GTK+ monitor the widget for motion and related events which are needed to determine when and where to show a tooltip.
- Connect to the "query-tooltip" signal. This signal will be emitted when a tooltip is supposed to be shown. One of the arguments passed to the signal handler is a GtkTooltip object. This is the object that we are about to display as a tooltip, and can be manipulated in your callback using functions like `gtk_tooltip_set_icon()`. There are functions for setting the tooltip's markup, setting an image from a named icon, or even putting in a custom widget.

  Return `TRUE` from your query-tooltip handler. This causes the tooltip to be show. If you return `FALSE`, it will not be shown.

In the probably rare case where you want to have even more control over the tooltip that is about to be shown, you can set your own GtkWindow which will be used as tooltip window. This works as follows:

- Set “has-tooltip” and connect to “query-tooltip” as before. Use `gtk_widget_set_tooltip_window()` to set a GtkWindow created by you as tooltip window.
- In the “query-tooltip” callback you can access your window using `gtk_widget_get_tooltip_window()` and manipulate as you wish. The semantics of the return value are exactly as before, return `TRUE` to show the window, `FALSE` to not show it.

Functions

gtk_tooltip_set_markup ()

```
void
gtk_tooltip_set_markup (GtkTooltip *tooltip,
                        const gchar *markup);
```

Sets the text of the tooltip to be `markup` , which is marked up with the Pango text markup language. If `markup` is `NULL`, the label will be hidden.

Parameters

tooltip	a GtkTooltip	
markup	a markup string (see Pango markup format) or `NULL`.	[allow-none]

Since: 2.12

---

gtk_tooltip_set_text ()

```
void
gtk_tooltip_set_text (GtkTooltip *tooltip,
                      const gchar *text);
```

Sets the text of the tooltip to be `text` . If `text` is `NULL`, the label will be hidden. See also `gtk_tooltip_set_markup()`.

Parameters

tooltip	a GtkTooltip	
text	a text string or `NULL`.	[allow-none]

Since: 2.12

---

gtk_tooltip_set_icon ()

```
void
gtk_tooltip_set_icon (GtkTooltip *tooltip,
                      GdkPixbuf *pixbuf);
```

Sets the icon of the tooltip (which is in front of the text) to be `pixbuf` . If `pixbuf` is `NULL`, the image will be hidden.

Parameters

tooltip	a GtkTooltip	
pixbuf	a GdkPixbuf, or `NULL`.	[allow-none]

Since: 2.12

---

gtk_tooltip_set_icon_from_stock ()

```
void
gtk_tooltip_set_icon_from_stock (GtkTooltip *tooltip,
                                 const gchar *stock_id,
                                 GtkIconSize size);
```

`gtk_tooltip_set_icon_from_stock` has been deprecated since version 3.10 and should not be used in newly-written code.

Use `gtk_tooltip_set_icon_from_icon_name()` instead.

Sets the icon of the tooltip (which is in front of the text) to be the stock item indicated by `stock_id` with the size indicated by `size` . If `stock_id` is `NULL`, the image will be hidden.

Parameters

tooltip	a GtkTooltip	
stock_id	a stock id, or `NULL`.	[allow-none]
size	a stock icon size (GtkIconSize).	[type int]

Since: 2.12

---

gtk_tooltip_set_icon_from_icon_name ()

```
void
gtk_tooltip_set_icon_from_icon_name (GtkTooltip *tooltip,
                                     const gchar *icon_name,
                                     GtkIconSize size);
```

Sets the icon of the tooltip (which is in front of the text) to be the icon indicated by `icon_name` with the size indicated by `size` . If `icon_name` is `NULL`, the image will be hidden.

Parameters

tooltip	a GtkTooltip	
icon_name	an icon name, or `NULL`.	[allow-none]
size	a stock icon size (GtkIconSize).	[type int]

Since: 2.14

---

gtk_tooltip_set_icon_from_gicon ()

```
void
gtk_tooltip_set_icon_from_gicon (GtkTooltip *tooltip,
                                 GIcon *gicon,
                                 GtkIconSize size);
```

Sets the icon of the tooltip (which is in front of the text) to be the icon indicated by `gicon` with the size indicated by `size` . If `gicon` is `NULL`, the image will be hidden.

Parameters

tooltip	a GtkTooltip	
gicon	a GIcon representing the icon, or `NULL`.	[allow-none]
size	a stock icon size (GtkIconSize).	[type int]

Since: 2.20

---

gtk_tooltip_set_custom ()

```
void
gtk_tooltip_set_custom (GtkTooltip *tooltip,
                        GtkWidget *custom_widget);
```

Replaces the widget packed into the tooltip with `custom_widget` . `custom_widget` does not get destroyed when the tooltip goes away. By default a box with a GtkImage and GtkLabel is embedded in the tooltip, which can be configured using `gtk_tooltip_set_markup()` and `gtk_tooltip_set_icon()`.

Parameters

tooltip	a GtkTooltip	
custom_widget	a GtkWidget, or `NULL` to unset the old custom widget.	[allow-none]

Since: 2.12

---

gtk_tooltip_trigger_tooltip_query ()

```
void
gtk_tooltip_trigger_tooltip_query (GdkDisplay *display);
```

Triggers a new tooltip query on `display` , in order to update the current visible tooltip, or to show/hide the current tooltip. This function is useful to call when, for example, the state of the widget changed by a key press.

Parameters

display	a GdkDisplay

Since: 2.12

---

gtk_tooltip_set_tip_area ()

```
void
gtk_tooltip_set_tip_area (GtkTooltip *tooltip,
                          const GdkRectangle *rect);
```

Sets the area of the widget, where the contents of this tooltip apply, to be `rect` (in widget coordinates). This is especially useful for properly setting tooltips on GtkTreeView rows and cells, GtkIconViews, etc.

For setting tooltips on GtkTreeView, please refer to the convenience functions for this: `gtk_tree_view_set_tooltip_row()` and `gtk_tree_view_set_tooltip_cell()`.

Parameters

tooltip	a GtkTooltip
rect	a GdkRectangle

Since: 2.12

Types and Values

GtkTooltip

```
typedef struct _GtkTooltip GtkTooltip;
```

GtkViewport

GtkViewport — An adapter which makes widgets scrollable

Functions

GtkWidget *	gtk_viewport_new ()
GtkAdjustment *	gtk_viewport_get_hadjustment ()
GtkAdjustment *	gtk_viewport_get_vadjustment ()
void	gtk_viewport_set_hadjustment ()
void	gtk_viewport_set_vadjustment ()
void	gtk_viewport_set_shadow_type ()
GtkShadowType	gtk_viewport_get_shadow_type ()
GdkWindow *	gtk_viewport_get_bin_window ()
GdkWindow *	gtk_viewport_get_view_window ()

Properties

GtkShadowType	shadow-type	Read / Write

Types and Values

struct	GtkViewport
struct	GtkViewportClass

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkContainer
            └── GtkBin
                └── GtkViewport
```

Implemented Interfaces

GtkViewport implements AtkImplementorIface, GtkBuildable and GtkScrollable.

Includes

```
#include <gtk/gtk.h>
```

Description

The GtkViewport widget acts as an adaptor class, implementing scrollability for child widgets that lack their own scrolling capabilities. Use GtkViewport to scroll child widgets such as GtkGrid, GtkBox, and so on.

If a widget has native scrolling abilities, such as GtkTextView, GtkTreeView or GtkIconView, it can be added to a GtkScrolledWindow with `gtk_container_add()`. If a widget does not, you must first add the widget to a GtkViewport, then add the viewport to the scrolled window. `gtk_container_add()` does this automatically if a child that does not implement GtkScrollable is added to a GtkScrolledWindow, so you can ignore the presence of the viewport.

The GtkViewport will start scrolling content only if allocated less than the child widget's minimum size in a given orientation.

CSS nodes

GtkViewport has a single CSS node with name viewport.

Functions

gtk_viewport_new ()

```
GtkWidget *
gtk_viewport_new (GtkAdjustment *hadjustment,
                  GtkAdjustment *vadjustment);
```

Creates a new GtkViewport with the given adjustments, or with default adjustments if none are given.

Parameters

hadjustment	horizontal adjustment.	[allow-none]
vadjustment	vertical adjustment.	[allow-none]

Returns

a new GtkViewport

---

gtk_viewport_get_hadjustment ()

```
GtkAdjustment *
gtk_viewport_get_hadjustment (GtkViewport *viewport);
```

`gtk_viewport_get_hadjustment` has been deprecated since version 3.0 and should not be used in newly-written code.

Use `gtk_scrollable_get_hadjustment()`

Returns the horizontal adjustment of the viewport.

Parameters

viewport	a GtkViewport.

Returns

the horizontal adjustment of `viewport` .

[transfer none]

---

gtk_viewport_get_vadjustment ()

```
GtkAdjustment *
gtk_viewport_get_vadjustment (GtkViewport *viewport);
```

`gtk_viewport_get_vadjustment` has been deprecated since version 3.0 and should not be used in newly-written code.

Use `gtk_scrollable_get_vadjustment()`

Returns the vertical adjustment of the viewport.

Parameters

viewport	a GtkViewport.

Returns

the vertical adjustment of `viewport` .

[transfer none]

---

gtk_viewport_set_hadjustment ()

```
void
gtk_viewport_set_hadjustment (GtkViewport *viewport,
                              GtkAdjustment *adjustment);
```

`gtk_viewport_set_hadjustment` has been deprecated since version 3.0 and should not be used in newly-written code.

Use `gtk_scrollable_set_hadjustment()`

Sets the horizontal adjustment of the viewport.

Parameters

viewport	a GtkViewport.	
adjustment	a GtkAdjustment.	[allow-none]

---

gtk_viewport_set_vadjustment ()

```
void
gtk_viewport_set_vadjustment (GtkViewport *viewport,
                              GtkAdjustment *adjustment);
```

`gtk_viewport_set_vadjustment` has been deprecated since version 3.0 and should not be used in newly-written code.

Use `gtk_scrollable_set_vadjustment()`

Sets the vertical adjustment of the viewport.

Parameters

viewport	a GtkViewport.	
adjustment	a GtkAdjustment.	[allow-none]

---

gtk_viewport_set_shadow_type ()

```
void
gtk_viewport_set_shadow_type (GtkViewport *viewport,
                              GtkShadowType type);
```

Sets the shadow type of the viewport.

Parameters

viewport	a GtkViewport.
type	the new shadow type.

---

gtk_viewport_get_shadow_type ()

```
GtkShadowType
gtk_viewport_get_shadow_type (GtkViewport *viewport);
```

Gets the shadow type of the GtkViewport. See `gtk_viewport_set_shadow_type()`.

Parameters

viewport	a GtkViewport

Returns

the shadow type

---

gtk_viewport_get_bin_window ()

```
GdkWindow *
gtk_viewport_get_bin_window (GtkViewport *viewport);
```

Gets the bin window of the GtkViewport.

Parameters

viewport	a GtkViewport

Returns

a GdkWindow.

[transfer none]

Since: 2.20

---

gtk_viewport_get_view_window ()

```
GdkWindow *
gtk_viewport_get_view_window (GtkViewport *viewport);
```

Gets the view window of the GtkViewport.

Parameters

viewport	a GtkViewport

Returns

a GdkWindow.

[transfer none]

Since: 2.22

Types and Values

struct GtkViewport

```
struct GtkViewport;
```

---

struct GtkViewportClass

```
struct GtkViewportClass {
  GtkBinClass parent_class;
};
```

Members

Property Details

The “shadow-type” property

```
  “shadow-type”              GtkShadowType
```

Determines how the shadowed box around the viewport is drawn.

Flags: Read / Write

Default value: GTK_SHADOW_IN

See Also

GtkScrolledWindow, GtkAdjustment

GtkAccessible

GtkAccessible — Accessibility support for widgets

Functions

void	gtk_accessible_connect_widget_destroyed ()
GtkWidget *	gtk_accessible_get_widget ()
void	gtk_accessible_set_widget ()

Properties

GtkWidget *	widget	Read / Write

Types and Values

struct	GtkAccessible

Object Hierarchy

```
GObject
└── AtkObject
    └── GtkAccessible
```

Includes

```
#include <gtk/gtk.h>
```

Description

The GtkAccessible class is the base class for accessible implementations for GtkWidget subclasses. It is a thin wrapper around AtkObject, which adds facilities for associating a widget with its accessible object.

An accessible implementation for a third-party widget should derive from GtkAccessible and implement the suitable interfaces from ATK, such as AtkText or AtkSelection. To establish the connection between the widget class and its corresponding acccessible implementation, override the get_accessible vfunc in GtkWidgetClass.

Functions

gtk_accessible_connect_widget_destroyed ()

```
void
gtk_accessible_connect_widget_destroyed
                               (GtkAccessible *accessible);
```

`gtk_accessible_connect_widget_destroyed` has been deprecated since version 3.4 and should not be used in newly-written code.

Use `gtk_accessible_set_widget()` and its vfuncs.

This function specifies the callback function to be called when the widget corresponding to a GtkAccessible is destroyed.

Parameters

accessible	a GtkAccessible

---

gtk_accessible_get_widget ()

```
GtkWidget *
gtk_accessible_get_widget (GtkAccessible *accessible);
```

Gets the GtkWidget corresponding to the GtkAccessible. The returned widget does not have a reference added, so you do not need to unref it.

Parameters

accessible	a GtkAccessible

Returns

pointer to the GtkWidget corresponding to the GtkAccessible, or `NULL`.

[nullable][transfer none]

Since: 2.22

---

gtk_accessible_set_widget ()

```
void
gtk_accessible_set_widget (GtkAccessible *accessible,
                           GtkWidget *widget);
```

Sets the GtkWidget corresponding to the GtkAccessible.

`accessible` will not hold a reference to `widget` . It is the caller's responsibility to ensure that when `widget` is destroyed, the widget is unset by calling this function again with `widget` set to `NULL`.

Parameters

accessible	a GtkAccessible	
widget	a GtkWidget or `NULL` to unset.	[allow-none]

Since: 2.22

Types and Values

struct GtkAccessible

```
struct GtkAccessible;
```

Property Details

The “widget” property

```
  “widget”                   GtkWidget *
```

The widget referenced by this accessible.

Flags: Read / Write

Abstract Base Classes

GtkWidget — Base class for all widgets

GtkContainer — Base class for widgets which contain other widgets

GtkBin — A container with just one child

GtkMenuShell — A base class for menu objects

GtkRange — Base class for widgets which visualize an adjustment

GtkIMContext — Base class for input method contexts

GtkNativeDialog — Integrate with native dialogs

GtkWidget

GtkWidget — Base class for all widgets

Functions

void	(*GtkCallback) ()
GtkWidget *	gtk_widget_new ()
void	gtk_widget_destroy ()
gboolean	gtk_widget_in_destruction ()
void	gtk_widget_destroyed ()
void	gtk_widget_unparent ()
void	gtk_widget_show ()
void	gtk_widget_show_now ()
void	gtk_widget_hide ()
void	gtk_widget_show_all ()
void	gtk_widget_map ()
void	gtk_widget_unmap ()
void	gtk_widget_realize ()
void	gtk_widget_unrealize ()

void	gtk_widget_draw ()
void	gtk_widget_queue_draw ()
void	gtk_widget_queue_resize ()
void	gtk_widget_queue_resize_no_redraw ()
void	gtk_widget_queue_allocate ()
GdkFrameClock *	gtk_widget_get_frame_clock ()
gint	gtk_widget_get_scale_factor ()
gboolean	(*GtkTickCallback) ()
guint	gtk_widget_add_tick_callback ()
void	gtk_widget_remove_tick_callback ()
void	gtk_widget_size_request ()
void	gtk_widget_get_child_requisition ()
void	gtk_widget_size_allocate ()
void	gtk_widget_size_allocate_with_baseline ()
void	gtk_widget_add_accelerator ()
gboolean	gtk_widget_remove_accelerator ()
void	gtk_widget_set_accel_path ()
GList *	gtk_widget_list_accel_closures ()
gboolean	gtk_widget_can_activate_accel ()
gboolean	gtk_widget_event ()
gboolean	gtk_widget_activate ()
void	gtk_widget_reparent ()
gboolean	gtk_widget_intersect ()
gboolean	gtk_widget_is_focus ()
void	gtk_widget_grab_focus ()
void	gtk_widget_grab_default ()
void	gtk_widget_set_name ()
const gchar *	gtk_widget_get_name ()
void	gtk_widget_set_state ()
void	gtk_widget_set_sensitive ()
void	gtk_widget_set_parent ()
void	gtk_widget_set_parent_window ()
GdkWindow *	gtk_widget_get_parent_window ()
void	gtk_widget_set_events ()
gint	gtk_widget_get_events ()
void	gtk_widget_add_events ()
void	gtk_widget_set_device_events ()
GdkEventMask	gtk_widget_get_device_events ()
void	gtk_widget_add_device_events ()
void	gtk_widget_set_device_enabled ()
gboolean	gtk_widget_get_device_enabled ()
GtkWidget *	gtk_widget_get_toplevel ()
GtkWidget *	gtk_widget_get_ancestor ()
GdkVisual *	gtk_widget_get_visual ()
void	gtk_widget_set_visual ()
void	gtk_widget_get_pointer ()
gboolean	gtk_widget_is_ancestor ()
gboolean	gtk_widget_translate_coordinates ()
gboolean	gtk_widget_hide_on_delete ()
void	gtk_widget_set_style ()
void	gtk_widget_ensure_style ()
GtkStyle *	gtk_widget_get_style ()

void	gtk_widget_reset_rc_styles ()
GtkStyle *	gtk_widget_get_default_style ()
void	gtk_widget_set_direction ()
GtkTextDirection	gtk_widget_get_direction ()
void	gtk_widget_set_default_direction ()
GtkTextDirection	gtk_widget_get_default_direction ()
void	gtk_widget_shape_combine_region ()
void	gtk_widget_input_shape_combine_region ()
void	gtk_widget_path ()
void	gtk_widget_class_path ()
gchar *	gtk_widget_get_composite_name ()
void	gtk_widget_override_background_color ()
void	gtk_widget_override_color ()
void	gtk_widget_override_font ()
void	gtk_widget_override_symbolic_color ()
void	gtk_widget_override_cursor ()
void	gtk_widget_modify_style ()
GtkRcStyle *	gtk_widget_get_modifier_style ()
void	gtk_widget_modify_fg ()
void	gtk_widget_modify_bg ()
void	gtk_widget_modify_text ()
void	gtk_widget_modify_base ()
void	gtk_widget_modify_font ()
void	gtk_widget_modify_cursor ()
PangoContext *	gtk_widget_create_pango_context ()
PangoContext *	gtk_widget_get_pango_context ()
void	gtk_widget_set_font_options ()
const cairo_font_options_t *	gtk_widget_get_font_options ()
void	gtk_widget_set_font_map ()
PangoFontMap *	gtk_widget_get_font_map ()
PangoLayout *	gtk_widget_create_pango_layout ()
GdkPixbuf *	gtk_widget_render_icon ()
GdkPixbuf *	gtk_widget_render_icon_pixbuf ()
void	gtk_widget_pop_composite_child ()
void	gtk_widget_push_composite_child ()
void	gtk_widget_queue_draw_area ()
void	gtk_widget_queue_draw_region ()
void	gtk_widget_set_app_paintable ()
void	gtk_widget_set_double_buffered ()
void	gtk_widget_set_redraw_on_allocate ()
void	gtk_widget_set_composite_name ()
gboolean	gtk_widget_mnemonic_activate ()
void	gtk_widget_class_install_style_property ()
void	gtk_widget_class_install_style_property_parser ()
GParamSpec *	gtk_widget_class_find_style_property ()
GParamSpec **	gtk_widget_class_list_style_properties ()
cairo_region_t *	gtk_widget_region_intersect ()
gint	gtk_widget_send_expose ()
gboolean	gtk_widget_send_focus_change ()
void	gtk_widget_style_get ()
void	gtk_widget_style_get_property ()
void	gtk_widget_style_get_valist ()

void	gtk_widget_style_attach ()
void	gtk_widget_class_set_accessible_type ()
void	gtk_widget_class_set_accessible_role ()
AtkObject *	gtk_widget_get_accessible ()
gboolean	gtk_widget_child_focus ()
void	gtk_widget_child_notify ()
void	gtk_widget_freeze_child_notify ()
gboolean	gtk_widget_get_child_visible ()
GtkWidget *	gtk_widget_get_parent ()
GtkSettings *	gtk_widget_get_settings ()
GtkClipboard *	gtk_widget_get_clipboard ()
GdkDisplay *	gtk_widget_get_display ()
GdkWindow *	gtk_widget_get_root_window ()
GdkScreen *	gtk_widget_get_screen ()
gboolean	gtk_widget_has_screen ()
void	gtk_widget_get_size_request ()
void	gtk_widget_set_child_visible ()
void	gtk_widget_set_size_request ()
void	gtk_widget_thaw_child_notify ()
void	gtk_widget_set_no_show_all ()
gboolean	gtk_widget_get_no_show_all ()
GList *	gtk_widget_list_mnemonic_labels ()
void	gtk_widget_add_mnemonic_label ()
void	gtk_widget_remove_mnemonic_label ()
gboolean	gtk_widget_is_composited ()
void	gtk_widget_error_bell ()
gboolean	gtk_widget_keynav_failed ()
gchar *	gtk_widget_get_tooltip_markup ()
void	gtk_widget_set_tooltip_markup ()
gchar *	gtk_widget_get_tooltip_text ()
void	gtk_widget_set_tooltip_text ()
GtkWindow *	gtk_widget_get_tooltip_window ()
void	gtk_widget_set_tooltip_window ()
gboolean	gtk_widget_get_has_tooltip ()
void	gtk_widget_set_has_tooltip ()
void	gtk_widget_trigger_tooltip_query ()
GdkWindow *	gtk_widget_get_window ()
void	gtk_widget_register_window ()
void	gtk_widget_unregister_window ()
gboolean	gtk_cairo_should_draw_window ()
void	gtk_cairo_transform_to_window ()
int	gtk_widget_get_allocated_width ()
int	gtk_widget_get_allocated_height ()
void	gtk_widget_get_allocation ()
void	gtk_widget_set_allocation ()
int	gtk_widget_get_allocated_baseline ()
void	gtk_widget_get_allocated_size ()
void	gtk_widget_get_clip ()
void	gtk_widget_set_clip ()
gboolean	gtk_widget_get_app_paintable ()
gboolean	gtk_widget_get_can_default ()
void	gtk_widget_set_can_default ()

gboolean	gtk_widget_get_can_focus ()
void	gtk_widget_set_can_focus ()
gboolean	gtk_widget_get_focus_on_click ()
void	gtk_widget_set_focus_on_click ()
gboolean	gtk_widget_get_double_buffered ()
gboolean	gtk_widget_get_has_window ()
void	gtk_widget_set_has_window ()
gboolean	gtk_widget_get_sensitive ()
gboolean	gtk_widget_is_sensitive ()
GtkStateType	gtk_widget_get_state ()
gboolean	gtk_widget_get_visible ()
gboolean	gtk_widget_is_visible ()
void	gtk_widget_set_visible ()
void	gtk_widget_set_state_flags ()
void	gtk_widget_unset_state_flags ()
GtkStateFlags	gtk_widget_get_state_flags ()
gboolean	gtk_widget_has_default ()
gboolean	gtk_widget_has_focus ()
gboolean	gtk_widget_has_visible_focus ()
gboolean	gtk_widget_has_grab ()
gboolean	gtk_widget_has_rc_style ()
gboolean	gtk_widget_is_drawable ()
gboolean	gtk_widget_is_toplevel ()
void	gtk_widget_set_window ()
void	gtk_widget_set_receives_default ()
gboolean	gtk_widget_get_receives_default ()
void	gtk_widget_set_support_multidevice ()
gboolean	gtk_widget_get_support_multidevice ()
void	gtk_widget_set_realized ()
gboolean	gtk_widget_get_realized ()
void	gtk_widget_set_mapped ()
gboolean	gtk_widget_get_mapped ()
void	gtk_widget_get_requisition ()
gboolean	gtk_widget_device_is_shadowed ()
GdkModifierType	gtk_widget_get_modifier_mask ()
void	gtk_widget_insert_action_group ()
double	gtk_widget_get_opacity ()
void	gtk_widget_set_opacity ()
const gchar **	gtk_widget_list_action_prefixes ()
GActionGroup *	gtk_widget_get_action_group ()
GtkWidgetPath *	gtk_widget_get_path ()
GtkStyleContext *	gtk_widget_get_style_context ()
void	gtk_widget_reset_style ()
const char *	gtk_widget_class_get_css_name ()
void	gtk_widget_class_set_css_name ()
GtkRequisition *	gtk_requisition_new ()
GtkRequisition *	gtk_requisition_copy ()
void	gtk_requisition_free ()
void	gtk_widget_get_preferred_height ()
void	gtk_widget_get_preferred_width ()
void	gtk_widget_get_preferred_height_for_width ()
void	gtk_widget_get_preferred_width_for_height ()

void	gtk_widget_get_preferred_height_and_baseline_for_width ()
GtkSizeRequestMode	gtk_widget_get_request_mode ()
void	gtk_widget_get_preferred_size ()
gint	gtk_distribute_natural_allocation ()
GtkAlign	gtk_widget_get_halign ()
void	gtk_widget_set_halign ()
GtkAlign	gtk_widget_get_valign ()
GtkAlign	gtk_widget_get_valign_with_baseline ()
void	gtk_widget_set_valign ()
gint	gtk_widget_get_margin_left ()
void	gtk_widget_set_margin_left ()
gint	gtk_widget_get_margin_right ()
void	gtk_widget_set_margin_right ()
gint	gtk_widget_get_margin_start ()
void	gtk_widget_set_margin_start ()
gint	gtk_widget_get_margin_end ()
void	gtk_widget_set_margin_end ()
gint	gtk_widget_get_margin_top ()
void	gtk_widget_set_margin_top ()
gint	gtk_widget_get_margin_bottom ()
void	gtk_widget_set_margin_bottom ()
gboolean	gtk_widget_get_hexpand ()
void	gtk_widget_set_hexpand ()
gboolean	gtk_widget_get_hexpand_set ()
void	gtk_widget_set_hexpand_set ()
gboolean	gtk_widget_get_vexpand ()
void	gtk_widget_set_vexpand ()
gboolean	gtk_widget_get_vexpand_set ()
void	gtk_widget_set_vexpand_set ()
void	gtk_widget_queue_compute_expand ()
gboolean	gtk_widget_compute_expand ()
void	gtk_widget_init_template ()
void	gtk_widget_class_set_template ()
void	gtk_widget_class_set_template_from_resource ()
GObject *	gtk_widget_get_template_child ()
#define	gtk_widget_class_bind_template_child()
#define	gtk_widget_class_bind_template_child_internal()
#define	gtk_widget_class_bind_template_child_private()
#define	gtk_widget_class_bind_template_child_internal_private()
void	gtk_widget_class_bind_template_child_full ()
#define	gtk_widget_class_bind_template_callback()
void	gtk_widget_class_bind_template_callback_full ()
void	gtk_widget_class_set_connect_func ()

Properties

gboolean	app-paintable	Read / Write
gboolean	can-default	Read / Write
gboolean	can-focus	Read / Write

gboolean	composite-child	Read
gboolean	double-buffered	Read / Write
GdkEventMask	events	Read / Write
gboolean	expand	Read / Write
gboolean	focus-on-click	Read / Write
GtkAlign	halign	Read / Write
gboolean	has-default	Read / Write
gboolean	has-focus	Read / Write
gboolean	has-tooltip	Read / Write
gint	height-request	Read / Write
gboolean	hexpand	Read / Write
gboolean	hexpand-set	Read / Write
gboolean	is-focus	Read / Write
gint	margin	Read / Write
gint	margin-bottom	Read / Write
gint	margin-end	Read / Write
gint	margin-left	Read / Write
gint	margin-right	Read / Write
gint	margin-start	Read / Write
gint	margin-top	Read / Write
gchar *	name	Read / Write
gboolean	no-show-all	Read / Write
gdouble	opacity	Read / Write
GtkContainer *	parent	Read / Write
gboolean	receives-default	Read / Write
gint	scale-factor	Read
gboolean	sensitive	Read / Write
GtkStyle *	style	Read / Write
gchar *	tooltip-markup	Read / Write
gchar *	tooltip-text	Read / Write
GtkAlign	valign	Read / Write
gboolean	vexpand	Read / Write
gboolean	vexpand-set	Read / Write
gboolean	visible	Read / Write
gint	width-request	Read / Write
GdkWindow *	window	Read

Style Properties

gfloat	cursor-aspect-ratio	Read
GdkColor *	cursor-color	Read
gchar *	focus-line-pattern	Read
gint	focus-line-width	Read
gint	focus-padding	Read
gboolean	interior-focus	Read
GdkColor *	link-color	Read
gint	scroll-arrow-hlength	Read
gint	scroll-arrow-vlength	Read
GdkColor *	secondary-cursor-color	Read
gint	separator-height	Read
gint	separator-width	Read

gint	text-handle-height	Read
gint	text-handle-width	Read
GdkColor *	visited-link-color	Read
gboolean	wide-separators	Read
gboolean	window-dragging	Read

Signals

void	accel-closures-changed	
gboolean	button-press-event	Run Last
gboolean	button-release-event	Run Last
gboolean	can-activate-accel	Run Last
void	child-notify	No Hooks
void	composited-changed	Action
gboolean	configure-event	Run Last
gboolean	damage-event	Run Last
gboolean	delete-event	Run Last
void	destroy	No Hooks
gboolean	destroy-event	Run Last
void	direction-changed	Run First
void	drag-begin	Run Last
void	drag-data-delete	Run Last
void	drag-data-get	Run Last
void	drag-data-received	Run Last
gboolean	drag-drop	Run Last
void	drag-end	Run Last
gboolean	drag-failed	Run Last
void	drag-leave	Run Last
gboolean	drag-motion	Run Last
gboolean	draw	Run Last
gboolean	enter-notify-event	Run Last
gboolean	event	Run Last
void	event-after	
gboolean	focus	Run Last
gboolean	focus-in-event	Run Last
gboolean	focus-out-event	Run Last
gboolean	grab-broken-event	Run Last
void	grab-focus	Action
void	grab-notify	Run First
void	hide	Run First
void	hierarchy-changed	Run Last
gboolean	key-press-event	Run Last
gboolean	key-release-event	Run Last
gboolean	keynav-failed	Run Last
gboolean	leave-notify-event	Run Last
void	map	Run First
gboolean	map-event	Run Last
gboolean	mnemonic-activate	Run Last
gboolean	motion-notify-event	Run Last
void	move-focus	Action
void	parent-set	Run First

gboolean	popup-menu	Action
gboolean	property-notify-event	Run Last
gboolean	proximity-in-event	Run Last
gboolean	proximity-out-event	Run Last
gboolean	query-tooltip	Run Last
void	realize	Run First
void	screen-changed	Run Last
gboolean	scroll-event	Run Last
gboolean	selection-clear-event	Run Last
void	selection-get	Run Last
gboolean	selection-notify-event	Run Last
void	selection-received	Run Last
gboolean	selection-request-event	Run Last
void	show	Run First
gboolean	show-help	Action
void	size-allocate	Run First
void	state-changed	Run First
void	state-flags-changed	Run First
void	style-set	Run First
void	style-updated	Run First
gboolean	touch-event	Run Last
void	unmap	Run First
gboolean	unmap-event	Run Last
void	unrealize	Run Last
gboolean	visibility-notify-event	Run Last
gboolean	window-state-event	Run Last

Types and Values

	GtkWidget
struct	GtkWidgetClass
	GtkRequisition
typedef	GtkAllocation
enum	GtkWidgetHelpType
enum	GtkTextDirection
enum	GtkStateType
enum	GtkSizeRequestMode
struct	GtkRequestedSize
enum	GtkAlign

Object Hierarchy

```
GBoxed
└── GtkRequisition
GObject
└── GInitiallyUnowned
    └── GtkWidget
        ├── GtkContainer
        ├── GtkMisc
        ├── GtkCalendar
        ├── GtkCellView
```

```
├── GtkDrawingArea
├── GtkEntry
├── GtkGLArea
├── GtkRange
├── GtkSeparator
├── GtkHSV
├── GtkInvisible
├── GtkProgressBar
├── GtkSpinner
├── GtkSwitch
└── GtkLevelBar
```

Known Derived Interfaces

GtkWidget is required by GtkActionable, GtkAppChooser, GtkCellEditable and GtkToolShell.

Implemented Interfaces

GtkWidget implements AtkImplementorIface and GtkBuildable.

Includes

```
#include <gtk/gtk.h>
```

Description

GtkWidget is the base class all widgets in GTK+ derive from. It manages the widget lifecycle, states and style.

Height-for-width Geometry Management

GTK+ uses a height-for-width (and width-for-height) geometry management system. Height-for-width means that a widget can change how much vertical space it needs, depending on the amount of horizontal space that it is given (and similar for width-for-height). The most common example is a label that reflows to fill up the available width, wraps to fewer lines, and therefore needs less height.

Height-for-width geometry management is implemented in GTK+ by way of five virtual methods:

- `GtkWidgetClass.get_request_mode()`
- `GtkWidgetClass.get_preferred_width()`
- `GtkWidgetClass.get_preferred_height()`
- `GtkWidgetClass.get_preferred_height_for_width()`
- `GtkWidgetClass.get_preferred_width_for_height()`
- `GtkWidgetClass.get_preferred_height_and_baseline_for_width()`

There are some important things to keep in mind when implementing height-for-width and when using it in container implementations.

The geometry management system will query a widget hierarchy in only one orientation at a time. When widgets are initially queried for their minimum sizes it is generally done in two initial passes in the GtkSizeRequestMode chosen by the toplevel.

For example, when queried in the normal GTK_SIZE_REQUEST_HEIGHT_FOR_WIDTH mode: First, the default minimum and natural width for each widget in the interface will be computed using gtk_widget_get_preferred_width(). Because the preferred widths for each container depend on the preferred widths of their children, this information propagates up the hierarchy, and finally a minimum and natural width is determined for the entire toplevel. Next, the toplevel will use the minimum width to query for the minimum height contextual to that width using gtk_widget_get_preferred_height_for_width(), which will also be a highly recursive operation. The minimum height for the minimum width is normally used to set the minimum size constraint on the toplevel (unless gtk_window_set_geometry_hints() is explicitly used instead).

After the toplevel window has initially requested its size in both dimensions it can go on to allocate itself a reasonable size (or a size previously specified with gtk_window_set_default_size()). During the recursive allocation process it's important to note that request cycles will be recursively executed while container widgets allocate their children. Each container widget, once allocated a size, will go on to first share the space in one orientation among its children and then request each child's height for its target allocated width or its width for allocated height, depending. In this way a GtkWidget will typically be requested its size a number of times before actually being allocated a size. The size a widget is finally allocated can of course differ from the size it has requested. For this reason, GtkWidget caches a small number of results to avoid re-querying for the same sizes in one allocation cycle.

See GtkContainer's geometry management section to learn more about how height-for-width allocations are performed by container widgets.

If a widget does move content around to intelligently use up the allocated size then it must support the request in both GtkSizeRequestModes even if the widget in question only trades sizes in a single orientation.

For instance, a GtkLabel that does height-for-width word wrapping will not expect to have GtkWidgetClass.get_preferred_height() called because that call is specific to a width-for-height request. In this case the label must return the height required for its own minimum possible width. By following this rule any widget that handles height-for-width or width-for-height requests will always be allocated at least enough space to fit its own content.

Here are some examples of how a GTK_SIZE_REQUEST_HEIGHT_FOR_WIDTH widget generally deals with width-for-height requests, for GtkWidgetClass.get_preferred_height() it will do:

```
1                                       static void
2                                       foo_widget_get_preferred_height (GtkWidget
3                                       *widget,
4                                                                        gint
5                                       *min_height,
6                                                                        gint
7                                       *nat_height)
8                                       {
9                                          if (i_am_in_height_for_width_mode)
10                                           {
11                                             gint min_width, nat_width;
12
13                                             GTK_WIDGET_GET_CLASS (widget)-
14                                      >get_preferred_width (widget,
15
16                                      &min_width,
17
18                                      &nat_width);
19                                             GTK_WIDGET_GET_CLASS (widget)-
20                                      >get_preferred_height_for_width
21
22                                      (widget,
23
```

```
24                                                  min_width,
25
                                                    min_height,

                                                    nat_height);
                                                 }
                                               else
                                                 {
                                                    ... some widgets do both. For
                                               instance, if a GtkLabel is
                                                    rotated to 90 degrees it will return
                                               the minimum and
                                                    natural height for the rotated label
                                               here.
                                                 }
                                               }
```

And in `GtkWidgetClass.get_preferred_width_for_height()` it will simply return the minimum and natural width:

```
1                                                   static void
2                                                   foo_widget_get_preferred_width_for_height
3                                                   (GtkWidget *widget,
4
5                                                   gint for_height,
6
7                                                   gint *min_width,
8
9                                                   gint *nat_width)
10                                                  {
11                                                     if (i_am_in_height_for_width_mode)
12                                                       {
13                                                        GTK_WIDGET_GET_CLASS (widget)-
14                                                  >get_preferred_width (widget,
15
16                                                  min_width,
17
18                                                  nat_width);
19                                                    }
                                                     else
                                                       {
                                                          ... again if a widget is sometimes
                                                  operating in
                                                          width-for-height mode (like a rotated
                                                  GtkLabel) it can go
                                                          ahead and do its real width for
                                                  height calculation here.
                                                       }
                                                  }
```

Often a widget needs to get its own request during size request or allocation. For example, when computing height it may need to also compute width. Or when deciding how to use an allocation, the widget may need to know its natural size. In these cases, the widget should be careful to call its virtual methods directly, like this:

```
1                                                   GTK_WIDGET_GET_CLASS(widget)-
2                                                   >get_preferred_width (widget,
3
                                                    &min,

                                                    &natural);
```

It will not work to use the wrapper functions, such as `gtk_widget_get_preferred_width()` inside your own size request implementation. These return a request adjusted by GtkSizeGroup and by the `GtkWidgetClass.adjust_size_request()` virtual method. If a widget used the wrappers inside its virtual method implementations, then the adjustments (such as widget margins) would be applied twice. GTK+ therefore does not allow this and will warn if you try to do it.

Of course if you are getting the size request for another widget, such as a child of a container, you must use the wrapper APIs. Otherwise, you would not properly consider widget margins, GtkSizeGroup, and so forth.

Since 3.10 GTK+ also supports baseline vertical alignment of widgets. This means that widgets are positioned such that the typographical baseline of widgets in the same row are aligned. This happens if a widget supports baselines, has a vertical alignment of `GTK_ALIGN_BASELINE`, and is inside a container that supports baselines and has a natural "row" that it aligns to the baseline, or a baseline assigned to it by the grandparent.

Baseline alignment support for a widget is done by the `GtkWidgetClass.get_preferred_height_and_baseline_for_width()` virtual function. It allows you to report a baseline in combination with the minimum and natural height. If there is no baseline you can return -1 to indicate this. The default implementation of this virtual function calls into the `GtkWidgetClass.get_preferred_height()` and `GtkWidgetClass.get_preferred_height_for_width()`, so if baselines are not supported it doesn't need to be implemented.

If a widget ends up baseline aligned it will be allocated all the space in the parent as if it was `GTK_ALIGN_FILL`, but the selected baseline can be found via `gtk_widget_get_allocated_baseline()`. If this has a value other than -1 you need to align the widget such that the baseline appears at the position.

Style Properties

GtkWidget introduces "style properties" - these are basically object properties that are stored not on the object, but in the style object associated to the widget. Style properties are set in resource files. This mechanism is used for configuring such things as the location of the scrollbar arrows through the theme, giving theme authors more control over the look of applications without the need to write a theme engine in C.

Use `gtk_widget_class_install_style_property()` to install style properties for a widget class, `gtk_widget_class_find_style_property()` or `gtk_widget_class_list_style_properties()` to get information about existing style properties and `gtk_widget_style_get_property()`, `gtk_widget_style_get()` or `gtk_widget_style_get_valist()` to obtain the value of a style property.

GtkWidget as GtkBuildable

The GtkWidget implementation of the GtkBuildable interface supports a custom <accelerator> element, which has attributes named "key", "modifiers" and "signal" and allows to specify accelerators.

An example of a UI definition fragment specifying an accelerator:

```
1                                       <object class="GtkButton">
2                                         <accelerator key="q"
3                                       modifiers="GDK_CONTROL_MASK"
                                        signal="clicked"/>
                                        </object>
```

In addition to accelerators, GtkWidget also support a custom <accessible> element, which supports actions and relations. Properties on the accessible implementation of an object can be set by accessing the internal child "accessible" of a GtkWidget.

An example of a UI definition fragment specifying an accessible:

```
1                                       <object class="GtkLabel" id="label1"/>
2                                         <property name="label">I am a Label for a
3                                       Button</property>
4                                       </object>
```

```
<object class="GtkButton" id="button1">
  <accessibility>
    <action action_name="click"
translatable="yes">Click the button.</action>
    <relation target="label1" type="labelled-
by"/>
  </accessibility>
  <child internal-child="accessible">
    <object class="AtkObject" id="a11y-
button1">
      <property name="accessible-
name">Clickable Button</property>
    </object>
  </child>
</object>
```

Finally, GtkWidget allows style information such as style classes to be associated with widgets, using the custom <style> element:

```
<object class="GtkButton" id="button1">
  <style>
    <class name="my-special-button-class"/>
    <class name="dark-button"/>
  </style>
</object>
```

Building composite widgets from template XML

GtkWidget exposes some facilities to automate the procedure of creating composite widgets using GtkBuilder interface description language.

To create composite widgets with GtkBuilder XML, one must associate the interface description with the widget class at class initialization time using `gtk_widget_class_set_template()`.

The interface description semantics expected in composite template descriptions is slightly different from regular GtkBuilder XML.

Unlike regular interface descriptions, `gtk_widget_class_set_template()` will expect a <template> tag as a direct child of the toplevel <interface> tag. The <template> tag must specify the “class” attribute which must be the type name of the widget. Optionally, the “parent” attribute may be specified to specify the direct parent type of the widget type, this is ignored by the GtkBuilder but required for Glade to introspect what kind of properties and internal children exist for a given type when the actual type does not exist.

The XML which is contained inside the <template> tag behaves as if it were added to the <object> tag defining `widget` itself. You may set properties on `widget` by inserting <property> tags into the <template> tag, and also add <child> tags to add children and extend `widget` in the normal way you would with <object> tags.

Additionally, <object> tags can also be added before and after the initial <template> tag in the normal way, allowing one to define auxiliary objects which might be referenced by other widgets declared as children of the <template> tag.

An example of a GtkBuilder Template Definition:

```
<interface>
  <template class="FooWidget"
parent="GtkBox">
    <property
name="orientation">GTK_ORIENTATION_HORIZONTAL
</property>
    <property name="spacing">4</property>
```

```
    <child>
      <object class="GtkButton"
id="hello_button">
        <property name="label">Hello
World</property>
        <signal name="clicked"
handler="hello_button_clicked"
object="FooWidget" swapped="yes"/>
      </object>
    </child>
    <child>
      <object class="GtkButton"
id="goodbye_button">
        <property name="label">Goodbye
World</property>
      </object>
    </child>
  </template>
</interface>
```

Typically, you'll place the template fragment into a file that is bundled with your project, using GResource. In order to load the template, you need to call `gtk_widget_class_set_template_from_resource()` from the class initialization of your GtkWidget type:

```
static void
foo_widget_class_init (FooWidgetClass *klass)
{
  // ...

  gtk_widget_class_set_template_from_resource
(GTK_WIDGET_CLASS (klass),

"/com/example/ui/foowidget.ui");
}
```

You will also need to call `gtk_widget_init_template()` from the instance initialization function:

```
static void
foo_widget_init (FooWidget *self)
{
  // ...
  gtk_widget_init_template (GTK_WIDGET
(self));
}
```

You can access widgets defined in the template using the `gtk_widget_get_template_child()` function, but you will typically declare a pointer in the instance private data structure of your type using the same name as the widget in the template definition, and call `gtk_widget_class_bind_template_child_private()` with that name, e.g.

```
typedef struct {
  GtkWidget *hello_button;
  GtkWidget *goodbye_button;
} FooWidgetPrivate;

G_DEFINE_TYPE_WITH_PRIVATE (FooWidget,
foo_widget, GTK_TYPE_BOX)

static void
foo_widget_class_init (FooWidgetClass *klass)
{
  // ...
  gtk_widget_class_set_template_from_resource
(GTK_WIDGET_CLASS (klass),

"/com/example/ui/foowidget.ui");

```

```
gtk_widget_class_bind_template_child_private
(GTK_WIDGET_CLASS (klass),

FooWidget, hello_button);

gtk_widget_class_bind_template_child_private
(GTK_WIDGET_CLASS (klass),

FooWidget, goodbye_button);
}

static void
foo_widget_init (FooWidget *widget)
{

}
```

You can also use gtk_widget_class_bind_template_callback() to connect a signal callback defined in the template with a function visible in the scope of the class, e.g.

```
// the signal handler has the instance and
user data swapped
// because of the swapped="yes" attribute in
the template XML
static void
hello_button_clicked (FooWidget *self,
                      GtkButton *button)
{
  g_print ("Hello, world!\n");
}

static void
foo_widget_class_init (FooWidgetClass *klass)
{
  // ...
  gtk_widget_class_set_template_from_resource
(GTK_WIDGET_CLASS (klass),

"/com/example/ui/foowidget.ui");
  gtk_widget_class_bind_template_callback
(GTK_WIDGET_CLASS (klass),
hello_button_clicked);
}
```

Functions

GtkCallback ()

```
void
(*GtkCallback) (GtkWidget *widget,
                gpointer data);
```

The type of the callback functions used for e.g. iterating over the children of a container, see gtk_container_foreach().

Parameters

widget	the widget to operate on	
data	user-supplied data.	[closure]

gtk_widget_new ()

```
GtkWidget *
gtk_widget_new (GType type,
                const gchar *first_property_name,
                ...);
```

This is a convenience function for creating a widget and setting its properties in one go. For example you might write: `gtk_widget_new (GTK_TYPE_LABEL, "label", "Hello World", "xalign", 0.0, NULL)` to create a left-aligned label. Equivalent to `g_object_new()`, but returns a widget so you don't have to cast the object yourself.

Parameters

type	type ID of the widget to create
first_property_name	name of first property to set
...	value of first property, followed by more properties, `NULL`-terminated

Returns

a new GtkWidget of type `widget_type`

---

gtk_widget_destroy ()

```
void
gtk_widget_destroy (GtkWidget *widget);
```

Destroys a widget.

When a widget is destroyed all references it holds on other objects will be released:

- if the widget is inside a container, it will be removed from its parent
- if the widget is a container, all its children will be destroyed, recursively
- if the widget is a top level, it will be removed from the list of top level widgets that GTK+ maintains internally

It's expected that all references held on the widget will also be released; you should connect to the "destroy" signal if you hold a reference to `widget` and you wish to remove it when this function is called. It is not necessary to do so if you are implementing a GtkContainer, as you'll be able to use the `GtkContainerClass.remove()` virtual function for that.

It's important to notice that `gtk_widget_destroy()` will only cause the `widget` to be finalized if no additional references, acquired using `g_object_ref()`, are held on it. In case additional references are in place, the `widget` will be in an "inert" state after calling this function; `widget` will still point to valid memory, allowing you to release the references you hold, but you may not query the widget's own state.

You should typically call this function on top level widgets, and rarely on child widgets.

See also: `gtk_container_remove()`

Parameters

widget	a GtkWidget	

---

gtk_widget_in_destruction ()

```
gboolean
gtk_widget_in_destruction (GtkWidget *widget);
```

Returns whether the widget is currently being destroyed. This information can sometimes be used to avoid doing unnecessary work.

Parameters

widget	a GtkWidget	

Returns

`TRUE` if `widget` is being destroyed

---

gtk_widget_destroyed ()

```
void
gtk_widget_destroyed (GtkWidget *widget,
                      GtkWidget **widget_pointer);
```

This function sets *`widget_pointer` to `NULL` if `widget_pointer` != `NULL`. It's intended to be used as a callback connected to the "destroy" signal of a widget. You connect `gtk_widget_destroyed()` as a signal handler, and pass the address of your widget variable as user data. Then when the widget is destroyed, the variable will be set to `NULL`. Useful for example to avoid multiple copies of the same dialog.

Parameters

widget	a GtkWidget	
widget_pointer	address of a variable that contains `widget`.	[inout][transfer none]

---

gtk_widget_unparent ()

```
void
gtk_widget_unparent (GtkWidget *widget);
```

This function is only for use in widget implementations. Should be called by implementations of the remove method on GtkContainer, to dissociate a child from the container.

Parameters

widget	a GtkWidget

---

gtk_widget_show ()

```
void
gtk_widget_show (GtkWidget *widget);
```

Flags a widget to be displayed. Any widget that isn't shown will not appear on the screen. If you want to show all the widgets in a container, it's easier to call `gtk_widget_show_all()` on the container, instead of individually showing the widgets.

Remember that you have to show the containers containing a widget, in addition to the widget itself, before it will appear onscreen.

When a toplevel container is shown, it is immediately realized and mapped; other shown widgets are realized and mapped when their toplevel container is realized and mapped.

Parameters

widget	a GtkWidget

---

gtk_widget_show_now ()

```
void
gtk_widget_show_now (GtkWidget *widget);
```

Shows a widget. If the widget is an unmapped toplevel widget (i.e. a GtkWindow that has not yet been shown), enter the main loop and wait for the window to actually be mapped. Be careful; because the main loop is running, anything can happen during this function.

Parameters

widget	a GtkWidget

---

gtk_widget_hide ()

```
void
gtk_widget_hide (GtkWidget *widget);
```

Reverses the effects of `gtk_widget_show()`, causing the widget to be hidden (invisible to the user).

Parameters

widget	a GtkWidget

---

gtk_widget_show_all ()

```
void
gtk_widget_show_all (GtkWidget *widget);
```

Recursively shows a widget, and any child widgets (if the widget is a container).

Parameters

widget	a GtkWidget

---

gtk_widget_map ()

```
void
gtk_widget_map (GtkWidget *widget);
```

This function is only for use in widget implementations. Causes a widget to be mapped if it isn't already.

Parameters

widget	a GtkWidget

---

gtk_widget_unmap ()

```
void
gtk_widget_unmap (GtkWidget *widget);
```

This function is only for use in widget implementations. Causes a widget to be unmapped if it's currently mapped.

Parameters

widget	a GtkWidget

---

gtk_widget_realize ()

```
void
gtk_widget_realize (GtkWidget *widget);
```

Creates the GDK (windowing system) resources associated with a widget. For example, `widget->window` will be created when a widget is realized. Normally realization happens implicitly; if you show a widget and all its parent containers, then the widget will be realized and mapped automatically.

Realizing a widget requires all the widget's parent widgets to be realized; calling `gtk_widget_realize()` realizes the widget's parents in addition to `widget` itself. If a widget is not yet inside a toplevel window when you realize it, bad things will happen.

This function is primarily used in widget implementations, and isn't very useful otherwise. Many times when

you think you might need it, a better approach is to connect to a signal that will be called after the widget is realized automatically, such as “draw”. Or simply `g_signal_connect()` to the “realize” signal.

Parameters

widget	a GtkWidget

---

gtk_widget_unrealize ()

```
void
gtk_widget_unrealize (GtkWidget *widget);
```

This function is only useful in widget implementations. Causes a widget to be unrealized (frees all GDK resources associated with the widget, such as `widget->window`).

Parameters

widget	a GtkWidget

---

gtk_widget_draw ()

```
void
gtk_widget_draw (GtkWidget *widget,
                 cairo_t *cr);
```

Draws `widget` to `cr` . The top left corner of the widget will be drawn to the currently set origin point of `cr` .

You should pass a cairo context as `cr` argument that is in an original state. Otherwise the resulting drawing is undefined. For example changing the operator using `cairo_set_operator()` or the line width using `cairo_set_line_width()` might have unwanted side effects. You may however change the context’s transform matrix - like with `cairo_scale()`, `cairo_translate()` or `cairo_set_matrix()` and clip region with `cairo_clip()` prior to calling this function. Also, it is fine to modify the context with `cairo_save()` and `cairo_push_group()` prior to calling this function.

Note that special-purpose widgets may contain special code for rendering to the screen and might appear differently on screen and when rendered using `gtk_widget_draw()`.

Parameters

widget	the widget to draw. It must be drawable (see `gtk_widget_is_drawable()`) and a size must have been allocated.
cr	a cairo context to draw to

Since: 3.0

---

gtk_widget_queue_draw ()

```
void
gtk_widget_queue_draw (GtkWidget *widget);
```

Equivalent to calling `gtk_widget_queue_draw_area()` for the entire area of a widget.

Parameters

widget	a GtkWidget

---

gtk_widget_queue_resize ()

```
void
gtk_widget_queue_resize (GtkWidget *widget);
```

This function is only for use in widget implementations. Flags a widget to have its size renegotiated; should be called when a widget for some reason has a new size request. For example, when you change the text in a GtkLabel, GtkLabel queues a resize to ensure there's enough space for the new text.

Note that you cannot call `gtk_widget_queue_resize()` on a widget from inside its implementation of the GtkWidgetClass::size_allocate virtual method. Calls to `gtk_widget_queue_resize()` from inside GtkWidgetClass::size_allocate will be silently ignored.

Parameters

widget	a GtkWidget

---

gtk_widget_queue_resize_no_redraw ()

```
void
gtk_widget_queue_resize_no_redraw (GtkWidget *widget);
```

This function works like `gtk_widget_queue_resize()`, except that the widget is not invalidated.

Parameters

widget	a GtkWidget

Since: 2.4

---

gtk_widget_queue_allocate ()

```
void
gtk_widget_queue_allocate (GtkWidget *widget);
```

This function is only for use in widget implementations.

Flags the widget for a rerun of the GtkWidgetClass::size_allocate function. Use this function instead of `gtk_widget_queue_resize()` when the `widget` 's size request didn't change but it wants to reposition its

contents.

An example user of this function is `gtk_widget_set_halign()`.

Parameters

widget	a GtkWidget

Since: 3.20

---

gtk_widget_get_frame_clock ()

```
GdkFrameClock *
gtk_widget_get_frame_clock (GtkWidget *widget);
```

Obtains the frame clock for a widget. The frame clock is a global “ticker” that can be used to drive animations and repaints. The most common reason to get the frame clock is to call `gdk_frame_clock_get_frame_time()`, in order to get a time to use for animating. For example you might record the start of the animation with an initial value from `gdk_frame_clock_get_frame_time()`, and then update the animation by calling `gdk_frame_clock_get_frame_time()` again during each repaint.

gdk_frame_clock_request_phase() will result in a new frame on the clock, but won’t necessarily repaint any widgets. To repaint a widget, you have to use `gtk_widget_queue_draw()` which invalidates the widget (thus scheduling it to receive a draw on the next frame). `gtk_widget_queue_draw()` will also end up requesting a frame on the appropriate frame clock.

A widget’s frame clock will not change while the widget is mapped. Reparenting a widget (which implies a temporary unmap) can change the widget’s frame clock.

Unrealized widgets do not have a frame clock.

Parameters

widget	a GtkWidget

Returns

a GdkFrameClock, or `NULL` if widget is unrealized.

[nullable][transfer none]

Since: 3.8

---

gtk_widget_get_scale_factor ()

```
gint
gtk_widget_get_scale_factor (GtkWidget *widget);
```

Retrieves the internal scale factor that maps from window coordinates to the actual device pixels. On traditional systems this is 1, on high density outputs, it can be a higher value (typically 2).

See `gdk_window_get_scale_factor()`.

Parameters

widget	a GtkWidget

Returns

the scale factor for `widget`

Since: 3.10

GtkTickCallback ()

```
gboolean
(*GtkTickCallback) (GtkWidget *widget,
                    GdkFrameClock *frame_clock,
                    gpointer user_data);
```

Callback type for adding a function to update animations. See `gtk_widget_add_tick_callback()`.

Parameters

widget	the widget
frame_clock	the frame clock for the widget (same as calling `gtk_widget_get_frame_clock()`)
user_data	user data passed to `gtk_widget_add_tick_callback()`.

Returns

`G_SOURCE_CONTINUE` if the tick callback should continue to be called, `G_SOURCE_REMOVE` if the tick callback should be removed.

Since: 3.8

---

gtk_widget_add_tick_callback ()

```
guint
gtk_widget_add_tick_callback (GtkWidget *widget,
                              GtkTickCallback callback,
                              gpointer user_data,
                              GDestroyNotify notify);
```

Queues an animation frame update and adds a callback to be called before each frame. Until the tick callback is removed, it will be called frequently (usually at the frame rate of the output device or as quickly as the application can be repainted, whichever is slower). For this reason, is most suitable for handling graphics that change every frame or every few frames. The tick callback does not automatically imply a relayout or repaint. If you want a repaint or relayout, and aren’t changing widget properties that would trigger that (for example,

changing the text of a GtkLabel), then you will have to call `gtk_widget_queue_resize()` or `gtk_widget_queue_draw_area()` yourself.

gdk_frame_clock_get_frame_time() should generally be used for timing continuous animations and `gdk_frame_timings_get_predicted_presentation_time()` if you are trying to display isolated frames at particular times.

This is a more convenient alternative to connecting directly to the "update" signal of GdkFrameClock, since you don't have to worry about when a GdkFrameClock is assigned to a widget.

Parameters

widget	a GtkWidget
callback	function to call for updating animations
user_data	data to pass to `callback`
notify	function to call to free `user_data` when the callback is removed.

Returns

an id for the connection of this callback. Remove the callback by passing it to `gtk_widget_remove_tick_callback()`

Since: 3.8

---

gtk_widget_remove_tick_callback ()

```
void
gtk_widget_remove_tick_callback (GtkWidget *widget,
                                 guint id);
```

Removes a tick callback previously registered with `gtk_widget_add_tick_callback()`.

Parameters

widget	a GtkWidget
id	an id returned by `gtk_widget_add_tick_callback()`

Since: 3.8

---

gtk_widget_size_request ()

```
void
gtk_widget_size_request (GtkWidget *widget,
                         GtkRequisition *requisition);
```

`gtk_widget_size_request` has been deprecated since version 3.0 and should not be used in newly-written code.

Use gtk_widget_get_preferred_size() instead.

This function is typically used when implementing a GtkContainer subclass. Obtains the preferred size of a widget. The container uses this information to arrange its child widgets and decide what size allocations to give them with gtk_widget_size_allocate().

You can also call this function from an application, with some caveats. Most notably, getting a size request requires the widget to be associated with a screen, because font information may be needed. Multihead-aware applications should keep this in mind.

Also remember that the size request is not necessarily the size a widget will actually be allocated.

Parameters

widget	a GtkWidget	
requisition	a GtkRequisition to be filled in.	[out]

---

gtk_widget_get_child_requisition ()

```
void
gtk_widget_get_child_requisition (GtkWidget *widget,
                                  GtkRequisition *requisition);
```

gtk_widget_get_child_requisition has been deprecated since version 3.0 and should not be used in newly-written code.

Use gtk_widget_get_preferred_size() instead.

This function is only for use in widget implementations. Obtains widget->requisition , unless someone has forced a particular geometry on the widget (e.g. with gtk_widget_set_size_request()), in which case it returns that geometry instead of the widget's requisition.

This function differs from gtk_widget_size_request() in that it retrieves the last size request value from widget->requisition , while gtk_widget_size_request() actually calls the "size_request" method on widget to compute the size request and fill in widget->requisition , and only then returns widget->requisition .

Because this function does not call the “size_request” method, it can only be used when you know that widget->requisition is up-to-date, that is, gtk_widget_size_request() has been called since the last time a resize was queued. In general, only container implementations have this information; applications should use gtk_widget_size_request().

Parameters

widget	a GtkWidget	
requisition	a GtkRequisition to be filled in.	[out]

---

gtk_widget_size_allocate ()

```
void
gtk_widget_size_allocate (GtkWidget *widget,
                          GtkAllocation *allocation);
```

This function is only used by GtkContainer subclasses, to assign a size and position to their child widgets.

In this function, the allocation may be adjusted. It will be forced to a 1x1 minimum size, and the adjust_size_allocation virtual method on the child will be used to adjust the allocation. Standard adjustments include removing the widget's margins, and applying the widget's "halign" and "valign" properties.

For baseline support in containers you need to use `gtk_widget_size_allocate_with_baseline()` instead.

Parameters

widget	a GtkWidget
allocation	position and size to be allocated to `widget`

---

gtk_widget_size_allocate_with_baseline ()

```
void
gtk_widget_size_allocate_with_baseline
                               (GtkWidget *widget,
                                GtkAllocation *allocation,
                                gint baseline);
```

This function is only used by GtkContainer subclasses, to assign a size, position and (optionally) baseline to their child widgets.

In this function, the allocation and baseline may be adjusted. It will be forced to a 1x1 minimum size, and the adjust_size_allocation virtual and adjust_baseline_allocation methods on the child will be used to adjust the allocation and baseline. Standard adjustments include removing the widget's margins, and applying the widget's "halign" and "valign" properties.

If the child widget does not have a valign of `GTK_ALIGN_BASELINE` the baseline argument is ignored and -1 is used instead.

Parameters

widget	a GtkWidget
allocation	position and size to be allocated to `widget`
baseline	The baseline of the child, or -1

Since: 3.10

---

gtk_widget_add_accelerator ()

```
void
gtk_widget_add_accelerator (GtkWidget *widget,
                            const gchar *accel_signal,
```

```
                                GtkAccelGroup *accel_group,
                                guint accel_key,
                                GdkModifierType accel_mods,
                                GtkAccelFlags accel_flags);
```

Installs an accelerator for this `widget` in `accel_group` that causes `accel_signal` to be emitted if the accelerator is activated. The `accel_group` needs to be added to the widget's toplevel via `gtk_window_add_accel_group()`, and the signal must be of type `G_SIGNAL_ACTION`. Accelerators added through this function are not user changeable during runtime. If you want to support accelerators that can be changed by the user, use `gtk_accel_map_add_entry()` and `gtk_widget_set_accel_path()` or `gtk_menu_item_set_accel_path()` instead.

Parameters

widget	widget to install an accelerator on
accel_signal	widget signal to emit on accelerator activation
accel_group	accel group for this widget, added to its toplevel
accel_key	GDK keyval of the accelerator
accel_mods	modifier key combination of the accelerator
accel_flags	flag accelerators, e.g. `GTK_ACCEL_VISIBLE`

---

gtk_widget_remove_accelerator ()

```
gboolean
gtk_widget_remove_accelerator (GtkWidget *widget,
                               GtkAccelGroup *accel_group,
                               guint accel_key,
                               GdkModifierType accel_mods);
```

Removes an accelerator from `widget` , previously installed with `gtk_widget_add_accelerator()`.

Parameters

widget	widget to install an accelerator on
accel_group	accel group for this widget
accel_key	GDK keyval of the accelerator
accel_mods	modifier key combination of the accelerator

Returns

whether an accelerator was installed and could be removed

---

gtk_widget_set_accel_path ()

```
void
gtk_widget_set_accel_path (GtkWidget *widget,
                           const gchar *accel_path,
                           GtkAccelGroup *accel_group);
```

Given an accelerator group, `accel_group` , and an accelerator path, `accel_path` , sets up an accelerator in `accel_group` so whenever the key binding that is defined for `accel_path` is pressed, `widget` will be activated. This removes any accelerators (for any accelerator group) installed by previous calls to `gtk_widget_set_accel_path()`. Associating accelerators with paths allows them to be modified by the user and the modifications to be saved for future use. (See `gtk_accel_map_save()`.)

This function is a low level function that would most likely be used by a menu creation system like GtkUIManager. If you use GtkUIManager, setting up accelerator paths will be done automatically.

Even when you you aren't using GtkUIManager, if you only want to set up accelerators on menu items `gtk_menu_item_set_accel_path()` provides a somewhat more convenient interface.

Note that `accel_path` string will be stored in a GQuark. Therefore, if you pass a static string, you can save some memory by interning it first with `g_intern_static_string()`.

Parameters

widget	a GtkWidget	
accel_path	path used to look up the accelerator.	[allow-none]
accel_group	a GtkAccelGroup.	[allow-none]

---

gtk_widget_list_accel_closures ()

```
GList *
gtk_widget_list_accel_closures (GtkWidget *widget);
```

Lists the closures used by `widget` for accelerator group connections with `gtk_accel_group_connect_by_path()` or `gtk_accel_group_connect()`. The closures can be used to monitor accelerator changes on `widget` , by connecting to the `GtkAccelGroup` ::accel-changed signal of the GtkAccelGroup of a closure which can be found out with `gtk_accel_group_from_accel_closure()`.

Parameters

widget	widget to list accelerator closures for

Returns

a newly allocated GList of closures.

[transfer container][element-type GClosure]

---

gtk_widget_can_activate_accel ()

```
gboolean
gtk_widget_can_activate_accel (GtkWidget *widget,
                               guint signal_id);
```

Determines whether an accelerator that activates the signal identified by `signal_id` can currently be activated. This is done by emitting the "can-activate-accel" signal on `widget` ; if the signal isn't overridden by a handler or in a derived widget, then the default check is that the widget must be sensitive, and the widget and all its ancestors mapped.

Parameters

widget	a GtkWidget
signal_id	the ID of a signal installed on `widget`

Returns

`TRUE` if the accelerator can be activated.

Since: 2.4

---

gtk_widget_event ()

```
gboolean
gtk_widget_event (GtkWidget *widget,
                  GdkEvent *event);
```

Rarely-used function. This function is used to emit the event signals on a widget (those signals should never be emitted without using this function to do so). If you want to synthesize an event though, don't use this function; instead, use `gtk_main_do_event()` so the event will behave as if it were in the event queue. Don't synthesize expose events; instead, use `gdk_window_invalidate_rect()` to invalidate a region of the window.

Parameters

widget	a GtkWidget
event	a GdkEvent

Returns

return from the event signal emission (`TRUE` if the event was handled)

---

gtk_widget_activate ()

```
gboolean
gtk_widget_activate (GtkWidget *widget);
```

For widgets that can be "activated" (buttons, menu items, etc.) this function activates them. Activation is what happens when you press Enter on a widget during key navigation. If `widget` isn't activatable, the function

returns FALSE.

Parameters

widget	a GtkWidget that's activatable

Returns

TRUE if the widget was activatable

---

gtk_widget_reparent ()

```
void
gtk_widget_reparent (GtkWidget *widget,
                     GtkWidget *new_parent);
```

gtk_widget_reparent has been deprecated since version 3.14 and should not be used in newly-written code.

Use gtk_container_remove() and gtk_container_add().

Moves a widget from one GtkContainer to another, handling reference count issues to avoid destroying the widget.

Parameters

widget	a GtkWidget
new_parent	a GtkContainer to move the widget into

---

gtk_widget_intersect ()

```
gboolean
gtk_widget_intersect (GtkWidget *widget,
                      const GdkRectangle *area,
                      GdkRectangle *intersection);
```

Computes the intersection of a widget 's area and area , storing the intersection in intersection , and returns TRUE if there was an intersection. intersection may be NULL if you're only interested in whether there was an intersection.

Parameters

widget	a GtkWidget	
area	a rectangle	
intersection	rectangle to store intersection of widget and area .	[out caller-allocates][optional]

Returns

TRUE if there was an intersection

---

gtk_widget_is_focus ()

```
gboolean
gtk_widget_is_focus (GtkWidget *widget);
```

Determines if the widget is the focus widget within its toplevel. (This does not mean that the “has-focus” property is necessarily set; “has-focus” will only be set if the toplevel widget additionally has the global input focus.)

Parameters

widget	a GtkWidget

Returns

TRUE if the widget is the focus widget.

---

gtk_widget_grab_focus ()

```
void
gtk_widget_grab_focus (GtkWidget *widget);
```

Causes `widget` to have the keyboard focus for the GtkWindow it's inside. `widget` must be a focusable widget, such as a GtkEntry; something like GtkFrame won’t work.

More precisely, it must have the `GTK_CAN_FOCUS` flag set. Use `gtk_widget_set_can_focus()` to modify that flag.

The widget also needs to be realized and mapped. This is indicated by the related signals. Grabbing the focus immediately after creating the widget will likely fail and cause critical warnings.

Parameters

widget	a GtkWidget

---

gtk_widget_grab_default ()

```
void
gtk_widget_grab_default (GtkWidget *widget);
```

Causes `widget` to become the default widget. `widget` must be able to be a default widget; typically you would ensure this yourself by calling `gtk_widget_set_can_default()` with a `TRUE` value. The default widget is activated when the user presses Enter in a window. Default widgets must be activatable, that is, `gtk_widget_activate()` should affect them. Note that GtkEntry widgets require the “activates-default”

property set to TRUE before they activate the default widget when Enter is pressed and the GtkEntry is focused.

Parameters

widget	a GtkWidget

---

gtk_widget_set_name ()

```
void
gtk_widget_set_name (GtkWidget *widget,
                     const gchar *name);
```

Widgets can be named, which allows you to refer to them from a CSS file. You can apply a style to widgets with a particular name in the CSS file. See the documentation for the CSS syntax (on the same page as the docs for GtkStyleContext).

Note that the CSS syntax has certain special characters to delimit and represent elements in a selector (period, #, >, *...), so using these will make your widget impossible to match by name. Any combination of alphanumeric symbols, dashes and underscores will suffice.

Parameters

widget	a GtkWidget
name	name for the widget

---

gtk_widget_get_name ()

```
const gchar *
gtk_widget_get_name (GtkWidget *widget);
```

Retrieves the name of a widget. See `gtk_widget_set_name()` for the significance of widget names.

Parameters

widget	a GtkWidget

Returns

name of the widget. This string is owned by GTK+ and should not be modified or freed

---

gtk_widget_set_state ()

```
void
gtk_widget_set_state (GtkWidget *widget,
                      GtkStateType state);
```

`gtk_widget_set_state` has been deprecated since version 3.0 and should not be used in newly-written code.

Use `gtk_widget_set_state_flags()` instead.

This function is for use in widget implementations. Sets the state of a widget (insensitive, prelighted, etc.) Usually you should set the state using wrapper functions such as `gtk_widget_set_sensitive()`.

Parameters

widget	a GtkWidget
state	new state for `widget`

---

gtk_widget_set_sensitive ()

```
void
gtk_widget_set_sensitive (GtkWidget *widget,
                          gboolean sensitive);
```

Sets the sensitivity of a widget. A widget is sensitive if the user can interact with it. Insensitive widgets are “grayed out” and the user can’t interact with them. Insensitive widgets are known as “inactive”, “disabled”, or “ghosted” in some other toolkits.

Parameters

widget	a GtkWidget
sensitive	`TRUE` to make the widget sensitive

---

gtk_widget_set_parent ()

```
void
gtk_widget_set_parent (GtkWidget *widget,
                       GtkWidget *parent);
```

This function is useful only when implementing subclasses of GtkContainer. Sets the container as the parent of `widget` , and takes care of some details such as updating the state and style of the child to reflect its new location. The opposite function is `gtk_widget_unparent()`.

Parameters

widget	a GtkWidget
parent	parent container

---

gtk_widget_set_parent_window ()

```
void
gtk_widget_set_parent_window (GtkWidget *widget,
                              GdkWindow *parent_window);
```

Sets a non default parent window for `widget` .

For GtkWindow classes, setting a `parent_window` effects whether the window is a toplevel window or can be embedded into other widgets.

For GtkWindow classes, this needs to be called before the window is realized.

Parameters

widget	a GtkWidget.
parent_window	the new parent window.

---

gtk_widget_get_parent_window ()

```
GdkWindow *
gtk_widget_get_parent_window (GtkWidget *widget);
```

Gets `widget` 's parent window, or `NULL` if it does not have one.

Parameters

widget	a GtkWidget.

Returns

the parent window of `widget` , or `NULL` if it does not have a parent window.

[transfer none][nullable]

---

gtk_widget_set_events ()

```
void
gtk_widget_set_events (GtkWidget *widget,
                       gint events);
```

Sets the event mask (see GdkEventMask) for a widget. The event mask determines which events a widget will receive. Keep in mind that different widgets have different default event masks, and by changing the event mask you may disrupt a widget's functionality, so be careful. This function must be called while a widget is unrealized. Consider `gtk_widget_add_events()` for widgets that are already realized, or if you want to preserve the existing event mask. This function can't be used with widgets that have no window. (See `gtk_widget_get_has_window()`). To get events on those widgets, place them inside a GtkEventBox and receive events on the event box.

Parameters

widget	a GtkWidget
events	event mask

---

gtk_widget_get_events ()

```
gint
gtk_widget_get_events (GtkWidget *widget);
```

Returns the event mask (see GdkEventMask) for the widget. These are the events that the widget will receive.

Note: Internally, the widget event mask will be the logical OR of the event mask set through `gtk_widget_set_events()` or `gtk_widget_add_events()`, and the event mask necessary to cater for every GtkEventController created for the widget.

Parameters

widget	a GtkWidget

Returns

event mask for `widget`

---

gtk_widget_add_events ()

```
void
gtk_widget_add_events (GtkWidget *widget,
                       gint events);
```

Adds the events in the bitfield `events` to the event mask for `widget` . See `gtk_widget_set_events()` and the input handling overview for details.

Parameters

widget	a GtkWidget
events	an event mask, see GdkEventMask

---

gtk_widget_set_device_events ()

```
void
gtk_widget_set_device_events (GtkWidget *widget,
                              GdkDevice *device,
                              GdkEventMask events);
```

Sets the device event mask (see GdkEventMask) for a widget. The event mask determines which events a widget will receive from `device` . Keep in mind that different widgets have different default event masks, and by changing the event mask you may disrupt a widget's functionality, so be careful. This function must be called while a widget is unrealized. Consider `gtk_widget_add_device_events()` for widgets that are already realized, or if you want to preserve the existing event mask. This function can't be used with windowless widgets (which return `FALSE` from `gtk_widget_get_has_window()`); to get events on those widgets, place them inside a GtkEventBox and receive events on the event box.

Parameters

widget	a GtkWidget
device	a GdkDevice
events	event mask

Since: 3.0

---

gtk_widget_get_device_events ()

```
GdkEventMask
gtk_widget_get_device_events (GtkWidget *widget,
                              GdkDevice *device);
```

Returns the events mask for the widget corresponding to an specific device. These are the events that the widget will receive when `device` operates on it.

Parameters

widget	a GtkWidget
device	a GdkDevice

Returns

device event mask for `widget`

Since: 3.0

---

gtk_widget_add_device_events ()

```
void
gtk_widget_add_device_events (GtkWidget *widget,
                              GdkDevice *device,
                              GdkEventMask events);
```

Adds the device events in the bitfield `events` to the event mask for `widget` . See `gtk_widget_set_device_events()` for details.

Parameters

widget	a GtkWidget
device	a GdkDevice
events	an event mask, see GdkEventMask

Since: 3.0

---

gtk_widget_set_device_enabled ()

```
void
gtk_widget_set_device_enabled (GtkWidget *widget,
```

```
                                GdkDevice *device,
                                gboolean enabled);
```

Enables or disables a GdkDevice to interact with `widget` and all its children.

It does so by descending through the GdkWindow hierarchy and enabling the same mask that is has for core events (i.e. the one that `gdk_window_get_events()` returns).

Parameters

widget	a GtkWidget
device	a GdkDevice
enabled	whether to enable the device

Since: 3.0

---

gtk_widget_get_device_enabled ()

```
gboolean
gtk_widget_get_device_enabled (GtkWidget *widget,
                               GdkDevice *device);
```

Returns whether `device` can interact with `widget` and its children. See `gtk_widget_set_device_enabled()`.

Parameters

widget	a GtkWidget
device	a GdkDevice

Returns

`TRUE` is `device` is enabled for `widget`

Since: 3.0

---

gtk_widget_get_toplevel ()

```
GtkWidget *
gtk_widget_get_toplevel (GtkWidget *widget);
```

This function returns the topmost widget in the container hierarchy `widget` is a part of. If `widget` has no parent widgets, it will be returned as the topmost widget. No reference will be added to the returned widget; it should not be unreferenced.

Note the difference in behavior vs. `gtk_widget_get_ancestor()`; `gtk_widget_get_ancestor (widget, GTK_TYPE_WINDOW)` would return `NULL` if `widget` wasn't inside a toplevel window, and if the window was inside a GtkWindow-derived widget which was in turn inside the toplevel GtkWindow. While the second case may seem unlikely, it actually happens when a GtkPlug is embedded inside a GtkSocket within the same application.

To reliably find the toplevel GtkWindow, use `gtk_widget_get_toplevel()` and call `GTK_IS_WINDOW()` on the result. For instance, to get the title of a widget's toplevel window, one might use:

```
1
2
3
4
5
6
7
8
9
10
11
```

```
static const char *
get_widget_toplevel_title (GtkWidget *widget)
{
  GtkWidget *toplevel =
gtk_widget_get_toplevel (widget);
  if (GTK_IS_WINDOW (toplevel))
    {
      return gtk_window_get_title (GTK_WINDOW
(toplevel));
    }

  return NULL;
}
```

Parameters

widget	a GtkWidget

Returns

the topmost ancestor of `widget` , or `widget` itself if there's no ancestor.

[transfer none]

---

gtk_widget_get_ancestor ()

```
GtkWidget *
gtk_widget_get_ancestor (GtkWidget *widget,
                         GType widget_type);
```

Gets the first ancestor of `widget` with type `widget_type` . For example, `gtk_widget_get_ancestor (widget, GTK_TYPE_BOX)` gets the first GtkBox that's an ancestor of `widget` . No reference will be added to the returned widget; it should not be unreferenced. See note about checking for a toplevel GtkWindow in the docs for `gtk_widget_get_toplevel()`.

Note that unlike `gtk_widget_is_ancestor()`, `gtk_widget_get_ancestor()` considers `widget` to be an ancestor of itself.

Parameters

widget	a GtkWidget
widget_type	ancestor type

Returns

the ancestor widget, or `NULL` if not found.

[transfer none][nullable]

---

gtk_widget_get_visual ()

```
GdkVisual *
gtk_widget_get_visual (GtkWidget *widget);
```

Gets the visual that will be used to render `widget` .

Parameters

widget	a GtkWidget

Returns

the visual for `widget` .

[transfer none]

---

gtk_widget_set_visual ()

```
void
gtk_widget_set_visual (GtkWidget *widget,
                       GdkVisual *visual);
```

Sets the visual that should be used for by widget and its children for creating GdkWindows. The visual must be on the same GdkScreen as returned by `gtk_widget_get_screen()`, so handling the "screen-changed" signal is necessary.

Setting a new `visual` will not cause `widget` to recreate its windows, so you should call this function before `widget` is realized.

Parameters

widget	a GtkWidget	
visual	visual to be used or `NULL` to unset a previous one.	[allow-none]

---

gtk_widget_get_pointer ()

```
void
gtk_widget_get_pointer (GtkWidget *widget,
                        gint *x,
                        gint *y);
```

`gtk_widget_get_pointer` has been deprecated since version 3.4 and should not be used in newly-written code.

Use `gdk_window_get_device_position()` instead.

Obtains the location of the mouse pointer in widget coordinates. Widget coordinates are a bit odd; for historical reasons, they are defined as `widget->window` coordinates for widgets that return `TRUE` for `gtk_widget_get_has_window()`; and are relative to `widget->allocation.x` , `widget->allocation.y` otherwise.

Parameters

widget	a GtkWidget	
x	return location for the X coordinate, or `NULL`.	[out][allow-none]
y	return location for the Y coordinate, or `NULL`.	[out][allow-none]

---

gtk_widget_is_ancestor ()

```
gboolean
gtk_widget_is_ancestor (GtkWidget *widget,
                        GtkWidget *ancestor);
```

Determines whether `widget` is somewhere inside `ancestor` , possibly with intermediate containers.

Parameters

widget	a GtkWidget
ancestor	another GtkWidget

Returns

`TRUE` if `ancestor` contains `widget` as a child, grandchild, great grandchild, etc.

---

gtk_widget_translate_coordinates ()

```
gboolean
gtk_widget_translate_coordinates (GtkWidget *src_widget,
                                  GtkWidget *dest_widget,
                                  gint src_x,
                                  gint src_y,
                                  gint *dest_x,
                                  gint *dest_y);
```

Translate coordinates relative to `src_widget` 's allocation to coordinates relative to `dest_widget` 's allocations. In order to perform this operation, both widgets must be realized, and must share a common toplevel.

Parameters

src_widget	a GtkWidget	
dest_widget	a GtkWidget	
src_x	X position relative to `src_widget`	
src_y	Y position relative to `src_widget`	
dest_x	location to store X position relative to `dest_widget` .	[out][optional]
dest_y	location to store Y position relative	[out][optional]

to dest_widget .

Returns

FALSE if either widget was not realized, or there was no common ancestor. In this case, nothing is stored in *dest_x and *dest_y . Otherwise TRUE.

---

gtk_widget_hide_on_delete ()

```
gboolean
gtk_widget_hide_on_delete (GtkWidget *widget);
```

Utility function; intended to be connected to the "delete-event" signal on a GtkWindow. The function calls gtk_widget_hide() on its argument, then returns TRUE. If connected to ::delete-event, the result is that clicking the close button for a window (on the window frame, top right corner usually) will hide but not destroy the window. By default, GTK+ destroys windows when ::delete-event is received.

Parameters

widget	a GtkWidget	

Returns

TRUE

---

gtk_widget_set_style ()

```
void
gtk_widget_set_style (GtkWidget *widget,
                      GtkStyle *style);
```

gtk_widget_set_style has been deprecated since version 3.0 and should not be used in newly-written code.

Use GtkStyleContext instead

Used to set the GtkStyle for a widget (widget->style). Since GTK 3, this function does nothing, the passed in style is ignored.

Parameters

widget	a GtkWidget	
style	a GtkStyle, or NULL to remove the effect of a previous call to gtk_widget_set_style() and go back to the default style.	[allow-none]

---

gtk_widget_ensure_style ()

```
void
gtk_widget_ensure_style (GtkWidget *widget);
```

`gtk_widget_ensure_style` has been deprecated since version 3.0 and should not be used in newly-written code.

Use GtkStyleContext instead

Ensures that `widget` has a style (`widget->style`).

Not a very useful function; most of the time, if you want the style, the widget is realized, and realized widgets are guaranteed to have a style already.

Parameters

widget	a GtkWidget

---

gtk_widget_get_style ()

```
GtkStyle *
gtk_widget_get_style (GtkWidget *widget);
```

`gtk_widget_get_style` has been deprecated since version 3.0 and should not be used in newly-written code.

Use GtkStyleContext instead

Simply an accessor function that returns `widget->style` .

Parameters

widget	a GtkWidget

Returns

the widget’s GtkStyle.

[transfer none]

---

gtk_widget_reset_rc_styles ()

```
void
gtk_widget_reset_rc_styles (GtkWidget *widget);
```

`gtk_widget_reset_rc_styles` has been deprecated since version 3.0 and should not be used in newly-written code.

Use GtkStyleContext instead, and `gtk_widget_reset_style()`

Reset the styles of `widget` and all descendents, so when they are looked up again, they get the correct values for the currently loaded RC file settings.

This function is not useful for applications.

Parameters

widget	a GtkWidget.

---

gtk_widget_get_default_style ()

```
GtkStyle *
gtk_widget_get_default_style (void);
```

`gtk_widget_get_default_style` has been deprecated since version 3.0 and should not be used in newly-written code.

Use GtkStyleContext instead, and `gtk_css_provider_get_default()` to obtain a GtkStyleProvider with the default widget style information.

Returns the default style used by all widgets initially.

Returns

the default style. This GtkStyle object is owned by GTK+ and should not be modified or freed.

[transfer none]

---

gtk_widget_set_direction ()

```
void
gtk_widget_set_direction (GtkWidget *widget,
                          GtkTextDirection dir);
```

Sets the reading direction on a particular widget. This direction controls the primary direction for widgets containing text, and also the direction in which the children of a container are packed. The ability to set the direction is present in order so that correct localization into languages with right-to-left reading directions can be done. Generally, applications will let the default reading direction present, except for containers where the containers are arranged in an order that is explicitly visual rather than logical (such as buttons for text justification).

If the direction is set to `GTK_TEXT_DIR_NONE`, then the value set by `gtk_widget_set_default_direction()` will be used.

Parameters

widget	a GtkWidget
dir	the new direction

---

gtk_widget_get_direction ()

```
GtkTextDirection
gtk_widget_get_direction (GtkWidget *widget);
```

Gets the reading direction for a particular widget. See `gtk_widget_set_direction()`.

Parameters

widget	a GtkWidget

Returns

the reading direction for the widget.

---

gtk_widget_set_default_direction ()

```
void
gtk_widget_set_default_direction (GtkTextDirection dir);
```

Sets the default reading direction for widgets where the direction has not been explicitly set by `gtk_widget_set_direction()`.

Parameters

dir	the new default direction. This cannot be `GTK_TEXT_DIR_NONE`.

---

gtk_widget_get_default_direction ()

```
GtkTextDirection
gtk_widget_get_default_direction (void);
```

Obtains the current default reading direction. See `gtk_widget_set_default_direction()`.

Returns

the current default direction.

---

gtk_widget_shape_combine_region ()

```
void
gtk_widget_shape_combine_region (GtkWidget *widget,
                                 cairo_region_t *region);
```

Sets a shape for this widget's GDK window. This allows for transparent windows etc., see `gdk_window_shape_combine_region()` for more information.

Parameters

widget	a GtkWidget	
region	shape to be added, or `NULL` to remove an existing shape.	[allow-none]

Since: 3.0

---

gtk_widget_input_shape_combine_region ()

```
void
gtk_widget_input_shape_combine_region (GtkWidget *widget,
                                       cairo_region_t *region);
```

Sets an input shape for this widget's GDK window. This allows for windows which react to mouse click in a nonrectangular region, see `gdk_window_input_shape_combine_region()` for more information.

Parameters

widget	a GtkWidget	
region	shape to be added, or `NULL` to remove an existing shape.	[allow-none]

Since: 3.0

---

gtk_widget_path ()

```
void
gtk_widget_path (GtkWidget *widget,
                 guint *path_length,
                 gchar **path,
                 gchar **path_reversed);
```

`gtk_widget_path` has been deprecated since version 3.0 and should not be used in newly-written code.

Use `gtk_widget_get_path()` instead

Obtains the full path to `widget` . The path is simply the name of a widget and all its parents in the container hierarchy, separated by periods. The name of a widget comes from `gtk_widget_get_name()`. Paths are used to apply styles to a widget in gtkrc configuration files. Widget names are the type of the widget by default (e.g. "GtkButton") or can be set to an application-specific value with `gtk_widget_set_name()`. By setting the name of a widget, you allow users or theme authors to apply styles to that specific widget in their gtkrc file. `path_reversed_p` fills in the path in reverse order, i.e. starting with `widget` 's name instead of starting with the name of `widget` 's outermost ancestor.

Parameters

widget	a GtkWidget	
path_length	location to store length of the path, or `NULL`.	[out][allow-none]
path	location to store allocated path string, or `NULL`.	[out][allow-none]

path_reversed	location to store allocated reverse path string, or `NULL`.	[out][allow-none]

gtk_widget_class_path ()

```
void
gtk_widget_class_path (GtkWidget *widget,
                       guint *path_length,
                       gchar **path,
                       gchar **path_reversed);
```

`gtk_widget_class_path` has been deprecated since version 3.0 and should not be used in newly-written code.

Use `gtk_widget_get_path()` instead

Same as `gtk_widget_path()`, but always uses the name of a widget's type, never uses a custom name set with `gtk_widget_set_name()`.

Parameters

widget	a GtkWidget	
path_length	location to store the length of the class path, or `NULL`.	[out][optional]
path	location to store the class path as an allocated string, or `NULL`.	[out][optional]
path_reversed	location to store the reverse class path as an allocated string, or `NULL`.	[out][optional]

gtk_widget_get_composite_name ()

```
gchar *
gtk_widget_get_composite_name (GtkWidget *widget);
```

`gtk_widget_get_composite_name` has been deprecated since version 3.10 and should not be used in newly-written code.

Use `gtk_widget_class_set_template()`, or don't use this API at all.

Obtains the composite name of a widget.

Parameters

widget	a GtkWidget

Returns

the composite name of `widget` , or `NULL` if `widget` is not a composite child. The string should be freed when it is no longer needed.

gtk_widget_override_background_color ()

```
void
gtk_widget_override_background_color (GtkWidget *widget,
                                      GtkStateFlags state,
                                      const GdkRGBA *color);
```

`gtk_widget_override_background_color` has been deprecated since version 3.16 and should not be used in newly-written code.

This function is not useful in the context of CSS-based rendering. If you wish to change the way a widget renders its background you should use a custom CSS style, through an application-specific GtkStyleProvider and a CSS style class. You can also override the default drawing of a widget through the "draw" signal, and use Cairo to draw a specific color, regardless of the CSS style.

Sets the background color to use for a widget.

All other style values are left untouched. See `gtk_widget_override_color()`.

Parameters

widget	a GtkWidget	
state	the state for which to set the background color	
color	the color to assign, or `NULL` to undo the effect of previous calls to `gtk_widget_override_background_color()`.	[allow-none]

Since: 3.0

---

gtk_widget_override_color ()

```
void
gtk_widget_override_color (GtkWidget *widget,
                           GtkStateFlags state,
                           const GdkRGBA *color);
```

`gtk_widget_override_color` has been deprecated since version 3.16 and should not be used in newly-written code.

Use a custom style provider and style classes instead

Sets the color to use for a widget.

All other style values are left untouched.

This function does not act recursively. Setting the color of a container does not affect its children. Note that some widgets that you may not think of as containers, for instance GtkButtons, are actually containers.

This API is mostly meant as a quick way for applications to change a widget appearance. If you are developing a widgets library and intend this change to be themeable, it is better done by setting meaningful CSS classes in your widget/container implementation through `gtk_style_context_add_class()`.

This way, your widget library can install a GtkCssProvider with the

GTK_STYLE_PROVIDER_PRIORITY_FALLBACK priority in order to provide a default styling for those widgets that need so, and this theming may fully overridden by the user's theme.

Note that for complex widgets this may bring in undesired results (such as uniform background color everywhere), in these cases it is better to fully style such widgets through a GtkCssProvider with the GTK_STYLE_PROVIDER_PRIORITY_APPLICATION priority.

Parameters

widget	a GtkWidget	
state	the state for which to set the color	
color	the color to assign, or NULL to undo the effect of previous calls to gtk_widget_override_color().	[allow-none]

Since: 3.0

---

gtk_widget_override_font ()

```
void
gtk_widget_override_font (GtkWidget *widget,
                          const PangoFontDescription *font_desc);
```

gtk_widget_override_font has been deprecated since version 3.16 and should not be used in newly-written code.

This function is not useful in the context of CSS-based rendering. If you wish to change the font a widget uses to render its text you should use a custom CSS style, through an application-specific GtkStyleProvider and a CSS style class.

Sets the font to use for a widget. All other style values are left untouched. See gtk_widget_override_color().

Parameters

widget	a GtkWidget	
font_desc	the font description to use, or NULL to undo the effect of previous calls to gtk_widget_override_font().	[allow-none]

Since: 3.0

---

gtk_widget_override_symbolic_color ()

```
void
gtk_widget_override_symbolic_color (GtkWidget *widget,
                                    const gchar *name,
                                    const GdkRGBA *color);
```

gtk_widget_override_symbolic_color has been deprecated since version 3.16 and should not be used in newly-written code.

This function is not useful in the context of CSS-based rendering. If you wish to change the color used to render symbolic icons you should use a custom CSS style, through an application-specific GtkStyleProvider and a CSS style class.

Sets a symbolic color for a widget.

All other style values are left untouched. See `gtk_widget_override_color()` for overriding the foreground or background color.

Parameters

widget	a GtkWidget	
name	the name of the symbolic color to modify	
color	the color to assign (does not need to be allocated), or `NULL` to undo the effect of previous calls to `gtk_widget_override_symbolic_color()`.	[allow-none]

Since: 3.0

---

gtk_widget_override_cursor ()

```
void
gtk_widget_override_cursor (GtkWidget *widget,
                            const GdkRGBA *cursor,
                            const GdkRGBA *secondary_cursor);
```

`gtk_widget_override_cursor` has been deprecated since version 3.16 and should not be used in newly-written code.

This function is not useful in the context of CSS-based rendering. If you wish to change the color used to render the primary and secondary cursors you should use a custom CSS style, through an application-specific GtkStyleProvider and a CSS style class.

Sets the cursor color to use in a widget, overriding the cursor-color and secondary-cursor-color style properties. All other style values are left untouched. See also `gtk_widget_modify_style()`.

Note that the underlying properties have the GdkColor type, so the alpha value in `primary` and `secondary` will be ignored.

Parameters

widget	a GtkWidget	
cursor	the color to use for primary cursor (does not need to be allocated), or `NULL` to undo the effect of previous calls to of `gtk_widget_override_cursor()`.	[allow-none]
secondary_cursor	the color to use for secondary cursor (does not need to be allocated), or	[allow-none]

	NULL to undo the effect of previous calls to of gtk_widget_override_cursor().

Since: 3.0

---

gtk_widget_modify_style ()

```
void
gtk_widget_modify_style (GtkWidget *widget,
                         GtkRcStyle *style);
```

gtk_widget_modify_style has been deprecated since version 3.0 and should not be used in newly-written code.

Use GtkStyleContext with a custom GtkStyleProvider instead

Modifies style values on the widget.

Modifications made using this technique take precedence over style values set via an RC file, however, they will be overridden if a style is explicitly set on the widget using gtk_widget_set_style(). The GtkRcStyle is designed so each field can either be set or unset, so it is possible, using this function, to modify some style values and leave the others unchanged.

Note that modifications made with this function are not cumulative with previous calls to gtk_widget_modify_style() or with such functions as gtk_widget_modify_fg(). If you wish to retain previous values, you must first call gtk_widget_get_modifier_style(), make your modifications to the returned style, then call gtk_widget_modify_style() with that style. On the other hand, if you first call gtk_widget_modify_style(), subsequent calls to such functions gtk_widget_modify_fg() will have a cumulative effect with the initial modifications.

Parameters

widget	a GtkWidget
style	the GtkRcStyle holding the style modifications

---

gtk_widget_get_modifier_style ()

```
GtkRcStyle *
gtk_widget_get_modifier_style (GtkWidget *widget);
```

gtk_widget_get_modifier_style has been deprecated since version 3.0 and should not be used in newly-written code.

Use GtkStyleContext with a custom GtkStyleProvider instead

Returns the current modifier style for the widget. (As set by gtk_widget_modify_style().) If no style has previously set, a new GtkRcStyle will be created with all values unset, and set as the modifier style for the widget. If you make changes to this rc style, you must call gtk_widget_modify_style(), passing in the returned rc style, to make sure that your changes take effect.

Caution: passing the style back to gtk_widget_modify_style() will normally end up destroying it, because

gtk_widget_modify_style() copies the passed-in style and sets the copy as the new modifier style, thus dropping any reference to the old modifier style. Add a reference to the modifier style if you want to keep it alive.

Parameters

widget	a GtkWidget

Returns

the modifier style for the widget. This rc style is owned by the widget. If you want to keep a pointer to value this around, you must add a refcount using g_object_ref().

[transfer none]

---

gtk_widget_modify_fg ()

```
void
gtk_widget_modify_fg (GtkWidget *widget,
                      GtkStateType state,
                      const GdkColor *color);
```

gtk_widget_modify_fg has been deprecated since version 3.0 and should not be used in newly-written code.

Use gtk_widget_override_color() instead

Sets the foreground color for a widget in a particular state.

All other style values are left untouched. See also gtk_widget_modify_style().

Parameters

widget	a GtkWidget	
state	the state for which to set the foreground color	
color	the color to assign (does not need to be allocated), or NULL to undo the effect of previous calls to of gtk_widget_modify_fg().	[allow-none]

---

gtk_widget_modify_bg ()

```
void
gtk_widget_modify_bg (GtkWidget *widget,
                      GtkStateType state,
                      const GdkColor *color);
```

gtk_widget_modify_bg has been deprecated since version 3.0 and should not be used in newly-written code.

Use gtk_widget_override_background_color() instead

Sets the background color for a widget in a particular state.

All other style values are left untouched. See also `gtk_widget_modify_style()`.

> Note that “no window” widgets (which have the `GTK_NO_WINDOW` flag set) draw on their parent container’s window and thus may not draw any background themselves. This is the case for e.g. GtkLabel.
>
> To modify the background of such widgets, you have to set the background color on their parent; if you want to set the background of a rectangular area around a label, try placing the label in a GtkEventBox widget and setting the background color on that.

Parameters

widget	a GtkWidget	
state	the state for which to set the background color	
color	the color to assign (does not need to be allocated), or `NULL` to undo the effect of previous calls to of `gtk_widget_modify_bg()`.	[allow-none]

---

gtk_widget_modify_text ()

```
void
gtk_widget_modify_text (GtkWidget *widget,
                        GtkStateType state,
                        const GdkColor *color);
```

`gtk_widget_modify_text` has been deprecated since version 3.0 and should not be used in newly-written code.

Use `gtk_widget_override_color()` instead

Sets the text color for a widget in a particular state.

All other style values are left untouched. The text color is the foreground color used along with the base color (see `gtk_widget_modify_base()`) for widgets such as GtkEntry and GtkTextView. See also `gtk_widget_modify_style()`.

Parameters

widget	a GtkWidget	
state	the state for which to set the text color	
color	the color to assign (does not need to be allocated), or `NULL` to undo the effect of previous calls to of `gtk_widget_modify_text()`.	[allow-none]

---

gtk_widget_modify_base ()

```
void
gtk_widget_modify_base (GtkWidget *widget,
                        GtkStateType state,
                        const GdkColor *color);
```

`gtk_widget_modify_base` has been deprecated since version 3.0 and should not be used in newly-written code.

Use `gtk_widget_override_background_color()` instead

Sets the base color for a widget in a particular state. All other style values are left untouched. The base color is the background color used along with the text color (see `gtk_widget_modify_text()`) for widgets such as GtkEntry and GtkTextView. See also `gtk_widget_modify_style()`.

> Note that “no window” widgets (which have the `GTK_NO_WINDOW` flag set) draw on their parent container’s window and thus may not draw any background themselves. This is the case for e.g. GtkLabel.
>
> To modify the background of such widgets, you have to set the base color on their parent; if you want to set the background of a rectangular area around a label, try placing the label in a GtkEventBox widget and setting the base color on that.

Parameters

widget	a GtkWidget	
state	the state for which to set the base color	
color	the color to assign (does not need to be allocated), or `NULL` to undo the effect of previous calls to of `gtk_widget_modify_base()`.	[allow-none]

---

gtk_widget_modify_font ()

```
void
gtk_widget_modify_font (GtkWidget *widget,
                        PangoFontDescription *font_desc);
```

`gtk_widget_modify_font` has been deprecated since version 3.0 and should not be used in newly-written code.

Use `gtk_widget_override_font()` instead

Sets the font to use for a widget.

All other style values are left untouched. See also `gtk_widget_modify_style()`.

Parameters

widget	a GtkWidget	
font_desc	the font description to use, or `NULL` to undo the effect of previous calls to `gtk_widget_modify_font()`.	[allow-none]

---

gtk_widget_modify_cursor ()

```
void
gtk_widget_modify_cursor (GtkWidget *widget,
                          const GdkColor *primary,
                          const GdkColor *secondary);
```

`gtk_widget_modify_cursor` has been deprecated since version 3.0 and should not be used in newly-written code.

Use `gtk_widget_override_cursor()` instead.

Sets the cursor color to use in a widget, overriding the GtkWidget cursor-color and secondary-cursor-color style properties.

All other style values are left untouched. See also `gtk_widget_modify_style()`.

Parameters

widget	a GtkWidget	
primary	the color to use for primary cursor (does not need to be allocated), or `NULL` to undo the effect of previous calls to of `gtk_widget_modify_cursor()`.	[nullable]
secondary	the color to use for secondary cursor (does not need to be allocated), or `NULL` to undo the effect of previous calls to of `gtk_widget_modify_cursor()`.	[nullable]

Since: 2.12

---

gtk_widget_create_pango_context ()

```
PangoContext *
gtk_widget_create_pango_context (GtkWidget *widget);
```

Creates a new PangoContext with the appropriate font map, font options, font description, and base direction for drawing text for this widget. See also `gtk_widget_get_pango_context()`.

Parameters

widget	a GtkWidget

Returns

the new PangoContext.

[transfer full]

---

gtk_widget_get_pango_context ()

```
PangoContext *
gtk_widget_get_pango_context (GtkWidget *widget);
```

Gets a PangoContext with the appropriate font map, font description, and base direction for this widget. Unlike the context returned by `gtk_widget_create_pango_context()`, this context is owned by the widget (it can be used until the screen for the widget changes or the widget is removed from its toplevel), and will be updated to match any changes to the widget's attributes. This can be tracked by using the "screen-changed" signal on the widget.

Parameters

widget	a GtkWidget	

Returns

the PangoContext for the widget.

[transfer none]

---

gtk_widget_set_font_options ()

```
void
gtk_widget_set_font_options (GtkWidget *widget,
                             const cairo_font_options_t *options);
```

Sets the cairo_font_options_t used for Pango rendering in this widget. When not set, the default font options for the GdkScreen will be used.

Parameters

widget	a GtkWidget	
options	a cairo_font_options_t, or `NULL` to unset any previously set default font options.	[allow-none]

Since: 3.18

---

gtk_widget_get_font_options ()

```
const cairo_font_options_t *
gtk_widget_get_font_options (GtkWidget *widget);
```

Returns the cairo_font_options_t used for Pango rendering. When not set, the defaults font options for the GdkScreen will be used.

Parameters

widget	a GtkWidget

Returns

the cairo_font_options_t or `NULL` if not set.

[transfer none][nullable]

Since: 3.18

---

gtk_widget_set_font_map ()

```
void
gtk_widget_set_font_map (GtkWidget *widget,
                         PangoFontMap *font_map);
```

Sets the font map to use for Pango rendering. When not set, the widget will inherit the font map from its parent.

Parameters

widget	a GtkWidget	
font_map	a PangoFontMap, or `NULL` to unset any previously set font map.	[allow-none]

Since: 3.18

---

gtk_widget_get_font_map ()

```
PangoFontMap *
gtk_widget_get_font_map (GtkWidget *widget);
```

Gets the font map that has been set with `gtk_widget_set_font_map()`.

Parameters

widget	a GtkWidget

Returns

A PangoFontMap, or `NULL`.

[transfer none][nullable]

Since: 3.18

---

gtk_widget_create_pango_layout ()

```
PangoLayout *
gtk_widget_create_pango_layout (GtkWidget *widget,
                                const gchar *text);
```

Creates a new PangoLayout with the appropriate font map, font description, and base direction for drawing text for this widget.

If you keep a PangoLayout created in this way around, you need to re-create it when the widget PangoContext is replaced. This can be tracked by using the "screen-changed" signal on the widget.

Parameters

widget	a GtkWidget	
text	text to set on the layout (can be `NULL`).	[nullable]

Returns

the new PangoLayout.

[transfer full]

---

gtk_widget_render_icon ()

```
GdkPixbuf *
gtk_widget_render_icon (GtkWidget *widget,
                        const gchar *stock_id,
                        GtkIconSize size,
                        const gchar *detail);
```

`gtk_widget_render_icon` has been deprecated since version 3.0 and should not be used in newly-written code.

Use `gtk_widget_render_icon_pixbuf()` instead.

A convenience function that uses the theme settings for `widget` to look up `stock_id` and render it to a pixbuf. `stock_id` should be a stock icon ID such as GTK_STOCK_OPEN or GTK_STOCK_OK. `size` should be a size such as GTK_ICON_SIZE_MENU. `detail` should be a string that identifies the widget or code doing the rendering, so that theme engines can special-case rendering for that widget or code.

The pixels in the returned GdkPixbuf are shared with the rest of the application and should not be modified. The pixbuf should be freed after use with `g_object_unref()`.

Parameters

widget	a GtkWidget	
stock_id	a stock ID	
size	a stock size (GtkIconSize). A size of `(GtkIconSize)-1` means render at the size of the source and don't scale (if there are multiple source sizes, GTK+ picks one of the	[type int]

	available sizes).	
detail	render detail to pass to theme engine.	[allow-none]

Returns

a new pixbuf, or `NULL` if the stock ID wasn't known.

[nullable][transfer full]

---

gtk_widget_render_icon_pixbuf ()

```
GdkPixbuf *
gtk_widget_render_icon_pixbuf (GtkWidget *widget,
                               const gchar *stock_id,
                               GtkIconSize size);
```

`gtk_widget_render_icon_pixbuf` has been deprecated since version 3.10 and should not be used in newly-written code.

Use `gtk_icon_theme_load_icon()` instead.

A convenience function that uses the theme engine and style settings for `widget` to look up `stock_id` and render it to a pixbuf. `stock_id` should be a stock icon ID such as GTK_STOCK_OPEN or GTK_STOCK_OK. `size` should be a size such as GTK_ICON_SIZE_MENU.

The pixels in the returned GdkPixbuf are shared with the rest of the application and should not be modified. The pixbuf should be freed after use with `g_object_unref()`.

Parameters

widget	a GtkWidget	
stock_id	a stock ID	
size	a stock size (GtkIconSize). A size of `(GtkIconSize)-1` means render at the size of the source and don't scale (if there are multiple source sizes, GTK+ picks one of the available sizes).	[type int]

Returns

a new pixbuf, or `NULL` if the stock ID wasn't known.

[transfer full][nullable]

Since: 3.0

---

gtk_widget_pop_composite_child ()

```
void
gtk_widget_pop_composite_child (void);
```

`gtk_widget_pop_composite_child` has been deprecated since version 3.10 and should not be used in newly-written code.

Use `gtk_widget_class_set_template()`, or don't use this API at all.

Cancels the effect of a previous call to `gtk_widget_push_composite_child()`.

---

gtk_widget_push_composite_child ()

```
void
gtk_widget_push_composite_child (void);
```

`gtk_widget_push_composite_child` has been deprecated since version 3.10 and should not be used in newly-written code.

This API never really worked well and was mostly unused, now we have a more complete mechanism for composite children, see `gtk_widget_class_set_template()`.

Makes all newly-created widgets as composite children until the corresponding `gtk_widget_pop_composite_child()` call.

A composite child is a child that's an implementation detail of the container it's inside and should not be visible to people using the container. Composite children aren't treated differently by GTK+ (but see `gtk_container_foreach()` vs. `gtk_container_forall()`), but e.g. GUI builders might want to treat them in a different way.

---

gtk_widget_queue_draw_area ()

```
void
gtk_widget_queue_draw_area (GtkWidget *widget,
                            gint x,
                            gint y,
                            gint width,
                            gint height);
```

Convenience function that calls `gtk_widget_queue_draw_region()` on the region created from the given coordinates.

The region here is specified in widget coordinates. Widget coordinates are a bit odd; for historical reasons, they are defined as `widget->window` coordinates for widgets that return `TRUE` for `gtk_widget_get_has_window()`, and are relative to `widget->allocation.x` , `widget->allocation.y` otherwise.

`width` or `height` may be 0, in this case this function does nothing. Negative values for `width` and `height` are not allowed.

Parameters

widget	a GtkWidget
x	x coordinate of upper-left corner of

	rectangle to redraw
y	y coordinate of upper-left corner of rectangle to redraw
width	width of region to draw
height	height of region to draw

---

gtk_widget_queue_draw_region ()

```
void
gtk_widget_queue_draw_region (GtkWidget *widget,
                              const cairo_region_t *region);
```

Invalidates the area of `widget` defined by `region` by calling `gdk_window_invalidate_region()` on the widget's window and all its child windows. Once the main loop becomes idle (after the current batch of events has been processed, roughly), the window will receive expose events for the union of all regions that have been invalidated.

Normally you would only use this function in widget implementations. You might also use it to schedule a redraw of a GtkDrawingArea or some portion thereof.

Parameters

widget	a GtkWidget
region	region to draw

Since: 3.0

---

gtk_widget_set_app_paintable ()

```
void
gtk_widget_set_app_paintable (GtkWidget *widget,
                              gboolean app_paintable);
```

Sets whether the application intends to draw on the widget in an "draw" handler.

This is a hint to the widget and does not affect the behavior of the GTK+ core; many widgets ignore this flag entirely. For widgets that do pay attention to the flag, such as GtkEventBox and GtkWindow, the effect is to suppress default themed drawing of the widget's background. (Children of the widget will still be drawn.) The application is then entirely responsible for drawing the widget background.

Note that the background is still drawn when the widget is mapped.

Parameters

widget	a GtkWidget
app_paintable	`TRUE` if the application will paint on the widget

---

gtk_widget_set_double_buffered ()

```
void
gtk_widget_set_double_buffered (GtkWidget *widget,
                                gboolean double_buffered);
```

`gtk_widget_set_double_buffered` has been deprecated since version 3.14 and should not be used in newly-written code.

This function does not work under non-X11 backends or with non-native windows. It should not be used in newly written code.

Widgets are double buffered by default; you can use this function to turn off the buffering. “Double buffered” simply means that `gdk_window_begin_draw_frame()` and `gdk_window_end_draw_frame()` are called automatically around expose events sent to the widget. `gdk_window_begin_draw_frame()` diverts all drawing to a widget's window to an offscreen buffer, and `gdk_window_end_draw_frame()` draws the buffer to the screen. The result is that users see the window update in one smooth step, and don’t see individual graphics primitives being rendered.

In very simple terms, double buffered widgets don’t flicker, so you would only use this function to turn off double buffering if you had special needs and really knew what you were doing.

Note: if you turn off double-buffering, you have to handle expose events, since even the clearing to the background color or pixmap will not happen automatically (as it is done in `gdk_window_begin_draw_frame()`).

In 3.10 GTK and GDK have been restructured for translucent drawing. Since then expose events for double-buffered widgets are culled into a single event to the toplevel GDK window. If you now unset double buffering, you will cause a separate rendering pass for every widget. This will likely cause rendering problems - in particular related to stacking - and usually increases rendering times significantly.

Parameters

widget	a GtkWidget
double_buffered	`TRUE` to double-buffer a widget

---

gtk_widget_set_redraw_on_allocate ()

```
void
gtk_widget_set_redraw_on_allocate (GtkWidget *widget,
                                   gboolean redraw_on_allocate);
```

Sets whether the entire widget is queued for drawing when its size allocation changes. By default, this setting is `TRUE` and the entire widget is redrawn on every size change. If your widget leaves the upper left unchanged when made bigger, turning this setting off will improve performance.

Note that for widgets where `gtk_widget_get_has_window()` is `FALSE` setting this flag to `FALSE` turns off all allocation on resizing: the widget will not even redraw if its position changes; this is to allow containers that don’t draw anything to avoid excess invalidations. If you set this flag on a widget with no window that does draw on `widget->window` , you are responsible for invalidating both the old and new allocation of the widget when the widget is moved and responsible for invalidating regions newly when the widget increases size.

Parameters

widget	a GtkWidget
redraw_on_allocate	if TRUE, the entire widget will be redrawn when it is allocated to a new size. Otherwise, only the new portion of the widget will be redrawn.

---

gtk_widget_set_composite_name ()

```
void
gtk_widget_set_composite_name (GtkWidget *widget,
                               const gchar *name);
```

gtk_widget_set_composite_name has been deprecated since version 3.10 and should not be used in newly-written code.

Use gtk_widget_class_set_template(), or don't use this API at all.

Sets a widgets composite name. The widget must be a composite child of its parent; see gtk_widget_push_composite_child().

Parameters

widget	a GtkWidget.
name	the name to set

---

gtk_widget_mnemonic_activate ()

```
gboolean
gtk_widget_mnemonic_activate (GtkWidget *widget,
                              gboolean group_cycling);
```

Emits the "mnemonic-activate" signal.

Parameters

widget	a GtkWidget
group_cycling	TRUE if there are other widgets with the same mnemonic

Returns

TRUE if the signal has been handled

---

gtk_widget_class_install_style_property ()

```
void
gtk_widget_class_install_style_property
                                (GtkWidgetClass *klass,
                                 GParamSpec *pspec);
```

Installs a style property on a widget class. The parser for the style property is determined by the value type of `pspec`.

Parameters

klass	a GtkWidgetClass
pspec	the GParamSpec for the property

---

gtk_widget_class_install_style_property_parser ()

```
void
gtk_widget_class_install_style_property_parser
                                (GtkWidgetClass *klass,
                                 GParamSpec *pspec,
                                 GtkRcPropertyParser parser);
```

Installs a style property on a widget class.

[skip]

Parameters

klass	a GtkWidgetClass
pspec	the GParamSpec for the style property
parser	the parser for the style property

---

gtk_widget_class_find_style_property ()

```
GParamSpec *
gtk_widget_class_find_style_property (GtkWidgetClass *klass,
                                      const gchar *property_name);
```

Finds a style property of a widget class by name.

Parameters

klass	a GtkWidgetClass
property_name	the name of the style property to find

Returns

the GParamSpec of the style property or `NULL` if `class` has no style property with that name.

[transfer none]

Since: 2.2

---

gtk_widget_class_list_style_properties ()

```
GParamSpec **
gtk_widget_class_list_style_properties
                                (GtkWidgetClass *klass,
                                 guint *n_properties);
```

Returns all style properties of a widget class.

Parameters

klass	a GtkWidgetClass	
n_properties	location to return the number of style properties found.	[out]

Returns

a newly allocated array of GParamSpec*. The array must be freed with `g_free()`.

[array length=n_properties][transfer container]

Since: 2.2

---

gtk_widget_region_intersect ()

```
cairo_region_t *
gtk_widget_region_intersect (GtkWidget *widget,
                             const cairo_region_t *region);
```

`gtk_widget_region_intersect` has been deprecated since version 3.14 and should not be used in newly-written code.

Use `gtk_widget_get_allocation()` and `cairo_region_intersect_rectangle()` to get the same behavior.

Computes the intersection of a `widget` 's area and `region` , returning the intersection. The result may be empty, use `cairo_region_is_empty()` to check.

Parameters

widget	a GtkWidget
region	a cairo_region_t, in the same coordinate system as `widget->allocation` . That is, relative to

	`widget->window` for widgets which return `FALSE` from `gtk_widget_get_has_window()`; relative to the parent window of `widget->window` otherwise.

Returns

A newly allocated region holding the intersection of `widget` and `region` .

---

gtk_widget_send_expose ()

```
gint
gtk_widget_send_expose (GtkWidget *widget,
                        GdkEvent *event);
```

`gtk_widget_send_expose` has been deprecated since version 3.22 and should not be used in newly-written code.

Application and widget code should not handle expose events directly; invalidation should use the GtkWidget API, and drawing should only happen inside "draw" implementations

Very rarely-used function. This function is used to emit an expose event on a widget. This function is not normally used directly. The only time it is used is when propagating an expose event to a windowless child widget (`gtk_widget_get_has_window()` is `FALSE`), and that is normally done using `gtk_container_propagate_draw()`.

If you want to force an area of a window to be redrawn, use `gdk_window_invalidate_rect()` or `gdk_window_invalidate_region()`. To cause the redraw to be done immediately, follow that call with a call to `gdk_window_process_updates()`.

Parameters

widget	a GtkWidget
event	a expose GdkEvent

Returns

return from the event signal emission (`TRUE` if the event was handled)

---

gtk_widget_send_focus_change ()

```
gboolean
gtk_widget_send_focus_change (GtkWidget *widget,
                              GdkEvent *event);
```

Sends the focus change `event` to `widget`

This function is not meant to be used by applications. The only time it should be used is when it is necessary for a GtkWidget to assign focus to a widget that is semantically owned by the first widget even though it's not a direct child - for instance, a search entry in a floating window similar to the quick search in GtkTreeView.

An example of its usage is:

```
GdkEvent *fevent = gdk_event_new
(GDK_FOCUS_CHANGE);

fevent->focus_change.type = GDK_FOCUS_CHANGE;
fevent->focus_change.in = TRUE;
fevent->focus_change.window =
_gtk_widget_get_window (widget);
if (fevent->focus_change.window != NULL)
  g_object_ref (fevent->focus_change.window);

gtk_widget_send_focus_change (widget,
fevent);

gdk_event_free (event);
```

Parameters

widget	a GtkWidget
event	a GdkEvent of type GDK_FOCUS_CHANGE

Returns

the return value from the event signal emission: `TRUE` if the event was handled, and `FALSE` otherwise

Since: 2.20

---

gtk_widget_style_get ()

```
void
gtk_widget_style_get (GtkWidget *widget,
                      const gchar *first_property_name,
                      ...);
```

Gets the values of a multiple style properties of `widget` .

Parameters

widget	a GtkWidget
first_property_name	the name of the first property to get
...	pairs of property names and locations to return the property values, starting with the location for `first_property_name` , terminated by `NULL`.

---

gtk_widget_style_get_property ()

```
void
gtk_widget_style_get_property (GtkWidget *widget,
```

```
                                const gchar *property_name,
                                GValue *value);
```

Gets the value of a style property of `widget`.

Parameters

widget	a GtkWidget
property_name	the name of a style property
value	location to return the property value

---

gtk_widget_style_get_valist ()

```
void
gtk_widget_style_get_valist (GtkWidget *widget,
                             const gchar *first_property_name,
                             va_list var_args);
```

Non-vararg variant of `gtk_widget_style_get()`. Used primarily by language bindings.

Parameters

widget	a GtkWidget
first_property_name	the name of the first property to get
var_args	a va_list of pairs of property names and locations to return the property values, starting with the location for `first_property_name`.

---

gtk_widget_style_attach ()

```
void
gtk_widget_style_attach (GtkWidget *widget);
```

`gtk_widget_style_attach` has been deprecated since version 3.0 and should not be used in newly-written code.

This step is unnecessary with GtkStyleContext.

This function attaches the widget's GtkStyle to the widget's GdkWindow. It is a replacement for

```
1                                       widget->style = gtk_style_attach (widget-
                                        >style, widget->window);
```

and should only ever be called in a derived widget's "realize" implementation which does not chain up to its parent class' "realize" implementation, because one of the parent classes (finally GtkWidget) would attach the style itself.

Parameters

widget	a GtkWidget

Since: 2.20

---

gtk_widget_class_set_accessible_type ()

```
void
gtk_widget_class_set_accessible_type (GtkWidgetClass *widget_class,
                                      GType type);
```

Sets the type to be used for creating accessibles for widgets of `widget_class` . The given `type` must be a subtype of the type used for accessibles of the parent class.

This function should only be called from class init functions of widgets.

Parameters

widget_class	class to set the accessible type for
type	The object type that implements the accessible for `widget_class`

Since: 3.2

---

gtk_widget_class_set_accessible_role ()

```
void
gtk_widget_class_set_accessible_role (GtkWidgetClass *widget_class,
                                      AtkRole role);
```

Sets the default AtkRole to be set on accessibles created for widgets of `widget_class` . Accessibles may decide to not honor this setting if their role reporting is more refined. Calls to `gtk_widget_class_set_accessible_type()` will reset this value.

In cases where you want more fine-grained control over the role of accessibles created for `widget_class` , you should provide your own accessible type and use `gtk_widget_class_set_accessible_type()` instead.

If `role` is ATK_ROLE_INVALID, the default role will not be changed and the accessible's default role will be used instead.

This function should only be called from class init functions of widgets.

Parameters

widget_class	class to set the accessible role for
role	The role to use for accessibles created for `widget_class`

Since: 3.2

---

gtk_widget_get_accessible ()

```
AtkObject *
gtk_widget_get_accessible (GtkWidget *widget);
```

Returns the accessible object that describes the widget to an assistive technology.

If accessibility support is not available, this AtkObject instance may be a no-op. Likewise, if no class-specific

AtkObject implementation is available for the widget instance in question, it will inherit an AtkObject implementation from the first ancestor class for which such an implementation is defined.

The documentation of the ATK library contains more information about accessible objects and their uses.

Parameters

widget	a GtkWidget

Returns

the AtkObject associated with `widget` .

[transfer none]

---

gtk_widget_child_focus ()

```
gboolean
gtk_widget_child_focus (GtkWidget *widget,
                        GtkDirectionType direction);
```

This function is used by custom widget implementations; if you're writing an app, you'd use `gtk_widget_grab_focus()` to move the focus to a particular widget, and `gtk_container_set_focus_chain()` to change the focus tab order. So you may want to investigate those functions instead.

gtk_widget_child_focus() is called by containers as the user moves around the window using keyboard shortcuts. `direction` indicates what kind of motion is taking place (up, down, left, right, tab forward, tab backward). `gtk_widget_child_focus()` emits the "focus" signal; widgets override the default handler for this signal in order to implement appropriate focus behavior.

The default ::focus handler for a widget should return `TRUE` if moving in `direction` left the focus on a focusable location inside that widget, and `FALSE` if moving in `direction` moved the focus outside the widget. If returning `TRUE`, widgets normally call `gtk_widget_grab_focus()` to place the focus accordingly; if returning `FALSE`, they don't modify the current focus location.

Parameters

widget	a GtkWidget
direction	direction of focus movement

Returns

`TRUE` if focus ended up inside `widget`

---

gtk_widget_child_notify ()

```
void
```

```
gtk_widget_child_notify (GtkWidget *widget,
                         const gchar *child_property);
```

Emits a "child-notify" signal for the child property `child_property` on `widget` .

This is the analogue of `g_object_notify()` for child properties.

Also see `gtk_container_child_notify()`.

Parameters

widget	a GtkWidget
child_property	the name of a child property installed on the class of `widget` 's parent

---

gtk_widget_freeze_child_notify ()

```
void
gtk_widget_freeze_child_notify (GtkWidget *widget);
```

Stops emission of "child-notify" signals on `widget` . The signals are queued until `gtk_widget_thaw_child_notify()` is called on `widget` .

This is the analogue of `g_object_freeze_notify()` for child properties.

Parameters

widget	a GtkWidget

---

gtk_widget_get_child_visible ()

```
gboolean
gtk_widget_get_child_visible (GtkWidget *widget);
```

Gets the value set with `gtk_widget_set_child_visible()`. If you feel a need to use this function, your code probably needs reorganization.

This function is only useful for container implementations and never should be called by an application.

Parameters

widget	a GtkWidget

Returns

`TRUE` if the widget is mapped with the parent.

---

gtk_widget_get_parent ()

```
GtkWidget *
gtk_widget_get_parent (GtkWidget *widget);
```

Returns the parent container of `widget` .

Parameters

widget	a GtkWidget

Returns

the parent container of `widget` , or `NULL`.

[transfer none][nullable]

---

gtk_widget_get_settings ()

```
GtkSettings *
gtk_widget_get_settings (GtkWidget *widget);
```

Gets the settings object holding the settings used for this widget.

Note that this function can only be called when the GtkWidget is attached to a toplevel, since the settings object is specific to a particular GdkScreen.

Parameters

widget	a GtkWidget

Returns

the relevant GtkSettings object.

[transfer none]

---

gtk_widget_get_clipboard ()

```
GtkClipboard *
gtk_widget_get_clipboard (GtkWidget *widget,
                          GdkAtom selection);
```

Returns the clipboard object for the given selection to be used with `widget` . `widget` must have a GdkDisplay associated with it, so must be attached to a toplevel window.

Parameters

widget	a GtkWidget
selection	a GdkAtom which identifies the

clipboard to use. `GDK_SELECTION_CLIPBOARD` gives the default clipboard. Another common value is `GDK_SELECTION_PRIMARY`, which gives the primary X selection.

Returns

the appropriate clipboard object. If no clipboard already exists, a new one will be created. Once a clipboard object has been created, it is persistent for all time.

[transfer none]

Since: 2.2

---

gtk_widget_get_display ()

```
GdkDisplay *
gtk_widget_get_display (GtkWidget *widget);
```

Get the GdkDisplay for the toplevel window associated with this widget. This function can only be called after the widget has been added to a widget hierarchy with a GtkWindow at the top.

In general, you should only create display specific resources when a widget has been realized, and you should free those resources when the widget is unrealized.

Parameters

widget	a GtkWidget

Returns

the GdkDisplay for the toplevel for this widget.

[transfer none]

Since: 2.2

---

gtk_widget_get_root_window ()

```
GdkWindow *
gtk_widget_get_root_window (GtkWidget *widget);
```

`gtk_widget_get_root_window` has been deprecated since version 3.12 and should not be used in newly-written code.

Use `gdk_screen_get_root_window()` instead

Get the root window where this widget is located. This function can only be called after the widget has been added to a widget hierarchy with GtkWindow at the top.

The root window is useful for such purposes as creating a popup GdkWindow associated with the window. In general, you should only create display specific resources when a widget has been realized, and you should free those resources when the widget is unrealized.

Parameters

widget	a GtkWidget

Returns

the GdkWindow root window for the toplevel for this widget.

[transfer none]

Since: 2.2

---

gtk_widget_get_screen ()

```
GdkScreen *
gtk_widget_get_screen (GtkWidget *widget);
```

Get the GdkScreen from the toplevel window associated with this widget. This function can only be called after the widget has been added to a widget hierarchy with a GtkWindow at the top.

In general, you should only create screen specific resources when a widget has been realized, and you should free those resources when the widget is unrealized.

Parameters

widget	a GtkWidget

Returns

the GdkScreen for the toplevel for this widget.

[transfer none]

Since: 2.2

---

gtk_widget_has_screen ()

```
gboolean
gtk_widget_has_screen (GtkWidget *widget);
```

Checks whether there is a GdkScreen is associated with this widget. All toplevel widgets have an associated screen, and all widgets added into a hierarchy with a toplevel window at the top.

Parameters

widget	a GtkWidget

Returns

`TRUE` if there is a GdkScreen associated with the widget.

Since: 2.2

---

gtk_widget_get_size_request ()

```
void
gtk_widget_get_size_request (GtkWidget *widget,
                             gint *width,
                             gint *height);
```

Gets the size request that was explicitly set for the widget using `gtk_widget_set_size_request()`. A value of -1 stored in `width` or `height` indicates that that dimension has not been set explicitly and the natural requisition of the widget will be used instead. See `gtk_widget_set_size_request()`. To get the size a widget will actually request, call `gtk_widget_get_preferred_size()` instead of this function.

Parameters

widget	a GtkWidget	
width	return location for width, or `NULL`.	[out][allow-none]
height	return location for height, or `NULL`.	[out][allow-none]

---

gtk_widget_set_child_visible ()

```
void
gtk_widget_set_child_visible (GtkWidget *widget,
                              gboolean is_visible);
```

Sets whether `widget` should be mapped along with its when its parent is mapped and `widget` has been shown with `gtk_widget_show()`.

The child visibility can be set for widget before it is added to a container with `gtk_widget_set_parent()`, to avoid mapping children unnecessary before immediately unmapping them. However it will be reset to its default state of `TRUE` when the widget is removed from a container.

Note that changing the child visibility of a widget does not queue a resize on the widget. Most of the time, the size of a widget is computed from all visible children, whether or not they are mapped. If this is not the case, the container can queue a resize itself.

This function is only useful for container implementations and never should be called by an application.

Parameters

widget	a GtkWidget

is_visible	if `TRUE`, `widget` should be mapped along with its parent.

---

gtk_widget_set_size_request ()

```
void
gtk_widget_set_size_request (GtkWidget *widget,
                             gint width,
                             gint height);
```

Sets the minimum size of a widget; that is, the widget's size request will be at least `width` by `height` . You can use this function to force a widget to be larger than it normally would be.

In most cases, `gtk_window_set_default_size()` is a better choice for toplevel windows than this function; setting the default size will still allow users to shrink the window. Setting the size request will force them to leave the window at least as large as the size request. When dealing with window sizes, `gtk_window_set_geometry_hints()` can be a useful function as well.

Note the inherent danger of setting any fixed size - themes, translations into other languages, different fonts, and user action can all change the appropriate size for a given widget. So, it's basically impossible to hardcode a size that will always be correct.

The size request of a widget is the smallest size a widget can accept while still functioning well and drawing itself correctly. However in some strange cases a widget may be allocated less than its requested size, and in many cases a widget may be allocated more space than it requested.

If the size request in a given direction is -1 (unset), then the "natural" size request of the widget will be used instead.

The size request set here does not include any margin from the GtkWidget properties margin-left, margin-right, margin-top, and margin-bottom, but it does include pretty much all other padding or border properties set by any subclass of GtkWidget.

Parameters

widget	a GtkWidget
width	width `widget` should request, or -1 to unset
height	height `widget` should request, or -1 to unset

---

gtk_widget_thaw_child_notify ()

```
void
gtk_widget_thaw_child_notify (GtkWidget *widget);
```

Reverts the effect of a previous call to `gtk_widget_freeze_child_notify()`. This causes all queued "child-notify" signals on `widget` to be emitted.

Parameters

widget	a GtkWidget

---

gtk_widget_set_no_show_all ()

```
void
gtk_widget_set_no_show_all (GtkWidget *widget,
                            gboolean no_show_all);
```

Sets the "no-show-all" property, which determines whether calls to `gtk_widget_show_all()` will affect this widget.

This is mostly for use in constructing widget hierarchies with externally controlled visibility, see GtkUIManager.

Parameters

widget	a GtkWidget
no_show_all	the new value for the "no-show-all" property

Since: 2.4

---

gtk_widget_get_no_show_all ()

```
gboolean
gtk_widget_get_no_show_all (GtkWidget *widget);
```

Returns the current value of the "no-show-all" property, which determines whether calls to `gtk_widget_show_all()` will affect this widget.

Parameters

widget	a GtkWidget

Returns

the current value of the "no-show-all" property.

Since: 2.4

---

gtk_widget_list_mnemonic_labels ()

```
GList *
gtk_widget_list_mnemonic_labels (GtkWidget *widget);
```

Returns a newly allocated list of the widgets, normally labels, for which this widget is the target of a mnemonic (see for example, `gtk_label_set_mnemonic_widget()`).

The widgets in the list are not individually referenced. If you want to iterate through the list and perform actions involving callbacks that might destroy the widgets, you must call `g_list_foreach (result, (GFunc)g_object_ref, NULL)` first, and then unref all the widgets afterwards.

Parameters

widget	a GtkWidget

Returns

the list of mnemonic labels; free this list with `g_list_free()` when you are done with it.

[element-type GtkWidget][transfer container]

Since: 2.4

---

gtk_widget_add_mnemonic_label ()

```
void
gtk_widget_add_mnemonic_label (GtkWidget *widget,
                               GtkWidget *label);
```

Adds a widget to the list of mnemonic labels for this widget. (See `gtk_widget_list_mnemonic_labels()`). Note the list of mnemonic labels for the widget is cleared when the widget is destroyed, so the caller must make sure to update its internal state at this point as well, by using a connection to the "destroy" signal or a weak notifier.

Parameters

widget	a GtkWidget
label	a GtkWidget that acts as a mnemonic label for `widget`

Since: 2.4

---

gtk_widget_remove_mnemonic_label ()

```
void
gtk_widget_remove_mnemonic_label (GtkWidget *widget,
                                  GtkWidget *label);
```

Removes a widget from the list of mnemonic labels for this widget. (See `gtk_widget_list_mnemonic_labels()`). The widget must have previously been added to the list with `gtk_widget_add_mnemonic_label()`.

Parameters

widget	a GtkWidget
label	a GtkWidget that was previously set

	as a mnemonic label for `widget` with `gtk_widget_add_mnemonic_label ()`.

Since: 2.4

---

gtk_widget_is_composited ()

```
gboolean
gtk_widget_is_composited (GtkWidget *widget);
```

`gtk_widget_is_composited` has been deprecated since version 3.22 and should not be used in newly-written code.

Use `gdk_screen_is_composited()` instead.

Whether `widget` can rely on having its alpha channel drawn correctly. On X11 this function returns whether a compositing manager is running for `widget` 's screen.

Please note that the semantics of this call will change in the future if used on a widget that has a composited window in its hierarchy (as set by `gdk_window_set_composited()`).

Parameters

widget	a GtkWidget

Returns

`TRUE` if the widget can rely on its alpha channel being drawn correctly.

Since: 2.10

---

gtk_widget_error_bell ()

```
void
gtk_widget_error_bell (GtkWidget *widget);
```

Notifies the user about an input-related error on this widget. If the "gtk-error-bell" setting is `TRUE`, it calls `gdk_window_beep()`, otherwise it does nothing.

Note that the effect of `gdk_window_beep()` can be configured in many ways, depending on the windowing backend and the desktop environment or window manager that is used.

Parameters

widget	a GtkWidget

Since: 2.12

---

gtk_widget_keynav_failed ()

```
gboolean
gtk_widget_keynav_failed (GtkWidget *widget,
                          GtkDirectionType direction);
```

This function should be called whenever keyboard navigation within a single widget hits a boundary. The function emits the "keynav-failed" signal on the widget and its return value should be interpreted in a way similar to the return value of `gtk_widget_child_focus()`:

When `TRUE` is returned, stay in the widget, the failed keyboard navigation is OK and/or there is nowhere we can/should move the focus to.

When `FALSE` is returned, the caller should continue with keyboard navigation outside the widget, e.g. by calling `gtk_widget_child_focus()` on the widget's toplevel.

The default ::keynav-failed handler returns `FALSE` for `GTK_DIR_TAB_FORWARD` and `GTK_DIR_TAB_BACKWARD`. For the other values of GtkDirectionType it returns `TRUE`.

Whenever the default handler returns `TRUE`, it also calls `gtk_widget_error_bell()` to notify the user of the failed keyboard navigation.

A use case for providing an own implementation of ::keynav-failed (either by connecting to it or by overriding it) would be a row of GtkEntry widgets where the user should be able to navigate the entire row with the cursor keys, as e.g. known from user interfaces that require entering license keys.

Parameters

widget	a GtkWidget
direction	direction of focus movement

Returns

`TRUE` if stopping keyboard navigation is fine, `FALSE` if the emitting widget should try to handle the keyboard navigation attempt in its parent container(s).

Since: 2.12

---

gtk_widget_get_tooltip_markup ()

```
gchar *
gtk_widget_get_tooltip_markup (GtkWidget *widget);
```

Gets the contents of the tooltip for `widget` .

Parameters

widget	a GtkWidget

Returns

the tooltip text, or `NULL`. You should free the returned string with `g_free()` when done.

[nullable]

Since: 2.12

---

gtk_widget_set_tooltip_markup ()

```
void
gtk_widget_set_tooltip_markup (GtkWidget *widget,
                               const gchar *markup);
```

Sets `markup` as the contents of the tooltip, which is marked up with the Pango text markup language.

This function will take care of setting “has-tooltip” to `TRUE` and of the default handler for the “query-tooltip” signal.

See also the “tooltip-markup” property and `gtk_tooltip_set_markup()`.

Parameters

widget	a GtkWidget	
markup	the contents of the tooltip for `widget` , or `NULL`.	[allow-none]

Since: 2.12

---

gtk_widget_get_tooltip_text ()

```
gchar *
gtk_widget_get_tooltip_text (GtkWidget *widget);
```

Gets the contents of the tooltip for `widget` .

Parameters

widget	a GtkWidget

Returns

the tooltip text, or `NULL`. You should free the returned string with `g_free()` when done.

[nullable]

Since: 2.12

---

gtk_widget_set_tooltip_text ()

```
void
gtk_widget_set_tooltip_text (GtkWidget *widget,
                             const gchar *text);
```

Sets `text` as the contents of the tooltip. This function will take care of setting “has-tooltip” to `TRUE` and of the default handler for the “query-tooltip” signal.

See also the “tooltip-text” property and `gtk_tooltip_set_text()`.

Parameters

widget	a GtkWidget	
text	the contents of the tooltip for `widget`.	[allow-none]

Since: 2.12

---

gtk_widget_get_tooltip_window ()

```
GtkWindow *
gtk_widget_get_tooltip_window (GtkWidget *widget);
```

Returns the GtkWindow of the current tooltip. This can be the GtkWindow created by default, or the custom tooltip window set using `gtk_widget_set_tooltip_window()`.

Parameters

widget	a GtkWidget	

Returns

The GtkWindow of the current tooltip.

[transfer none]

Since: 2.12

---

gtk_widget_set_tooltip_window ()

```
void
gtk_widget_set_tooltip_window (GtkWidget *widget,
                               GtkWindow *custom_window);
```

Replaces the default window used for displaying tooltips with `custom_window`. GTK+ will take care of showing and hiding `custom_window` at the right moment, to behave likewise as the default tooltip window. If `custom_window` is `NULL`, the default tooltip window will be used.

Parameters

widget	a GtkWidget	
custom_window	a GtkWindow, or `NULL`.	[allow-none]

Since: 2.12

---

gtk_widget_get_has_tooltip ()

```
gboolean
gtk_widget_get_has_tooltip (GtkWidget *widget);
```

Returns the current value of the has-tooltip property. See “has-tooltip” for more information.

Parameters

widget	a GtkWidget

Returns

current value of has-tooltip on `widget` .

Since: 2.12

---

gtk_widget_set_has_tooltip ()

```
void
gtk_widget_set_has_tooltip (GtkWidget *widget,
                            gboolean has_tooltip);
```

Sets the has-tooltip property on `widget` to `has_tooltip` . See “has-tooltip” for more information.

Parameters

widget	a GtkWidget
has_tooltip	whether or not `widget` has a tooltip.

Since: 2.12

---

gtk_widget_trigger_tooltip_query ()

```
void
gtk_widget_trigger_tooltip_query (GtkWidget *widget);
```

Triggers a tooltip query on the display where the toplevel of `widget` is located. See `gtk_tooltip_trigger_tooltip_query()` for more information.

Parameters

widget	a GtkWidget

Since: 2.12

---

gtk_widget_get_window ()

```
GdkWindow *
gtk_widget_get_window (GtkWidget *widget);
```

Returns the widget's window if it is realized, `NULL` otherwise

Parameters

widget	a GtkWidget

Returns

`widget` 's window.

[transfer none][nullable]

Since: 2.14

---

gtk_widget_register_window ()

```
void
gtk_widget_register_window (GtkWidget *widget,
                            GdkWindow *window);
```

Registers a GdkWindow with the widget and sets it up so that the widget receives events for it. Call `gtk_widget_unregister_window()` when destroying the window.

Before 3.8 you needed to call `gdk_window_set_user_data()` directly to set this up. This is now deprecated and you should use `gtk_widget_register_window()` instead. Old code will keep working as is, although some new features like transparency might not work perfectly.

Parameters

widget	a GtkWidget
window	a GdkWindow

Since: 3.8

---

gtk_widget_unregister_window ()

```
void
gtk_widget_unregister_window (GtkWidget *widget,
                              GdkWindow *window);
```

Unregisters a GdkWindow from the widget that was previously set up with `gtk_widget_register_window()`. You need to call this when the window is no longer used by the widget, such as when you destroy it.

Parameters

widget	a GtkWidget

window	a GdkWindow

Since: 3.8

---

gtk_cairo_should_draw_window ()

```
gboolean
gtk_cairo_should_draw_window (cairo_t *cr,
                              GdkWindow *window);
```

This function is supposed to be called in "draw" implementations for widgets that support multiple windows. `cr` must be untransformed from invoking of the draw function. This function will return `TRUE` if the contents of the given `window` are supposed to be drawn and `FALSE` otherwise. Note that when the drawing was not initiated by the windowing system this function will return `TRUE` for all windows, so you need to draw the bottommost window first. Also, do not use "else if" statements to check which window should be drawn.

Parameters

cr	a cairo context
window	the window to check. `window` may not be an input-only window.

Returns

`TRUE` if `window` should be drawn

Since: 3.0

---

gtk_cairo_transform_to_window ()

```
void
gtk_cairo_transform_to_window (cairo_t *cr,
                               GtkWidget *widget,
                               GdkWindow *window);
```

Transforms the given cairo context `cr` that from `widget` -relative coordinates to `window` -relative coordinates. If the `widget` 's window is not an ancestor of `window` , no modification will be applied.

This is the inverse to the transformation GTK applies when preparing an expose event to be emitted with the "draw" signal. It is intended to help porting multiwindow widgets from GTK+ 2 to the rendering architecture of GTK+ 3.

Parameters

cr	the cairo context to transform
widget	the widget the context is currently centered for
window	the window to transform the context to

Since: 3.0

---

gtk_widget_get_allocated_width ()

```
int
gtk_widget_get_allocated_width (GtkWidget *widget);
```

Returns the width that has currently been allocated to `widget` . This function is intended to be used when implementing handlers for the "draw" function.

Parameters

widget	the widget to query

Returns

the width of the `widget`

---

gtk_widget_get_allocated_height ()

```
int
gtk_widget_get_allocated_height (GtkWidget *widget);
```

Returns the height that has currently been allocated to `widget` . This function is intended to be used when implementing handlers for the "draw" function.

Parameters

widget	the widget to query

Returns

the height of the `widget`

---

gtk_widget_get_allocation ()

```
void
gtk_widget_get_allocation (GtkWidget *widget,
                           GtkAllocation *allocation);
```

Retrieves the widget's allocation.

Note, when implementing a GtkContainer: a widget's allocation will be its "adjusted" allocation, that is, the widget's parent container typically calls `gtk_widget_size_allocate()` with an allocation, and that allocation is then adjusted (to handle margin and alignment for example) before assignment to the widget. `gtk_widget_get_allocation()` returns the adjusted allocation that was actually assigned to the widget. The adjusted allocation is guaranteed to be completely contained within the `gtk_widget_size_allocate()` allocation, however. So a GtkContainer is guaranteed that its children stay inside the assigned bounds, but not that they have exactly the bounds the container assigned. There is no way to get the original allocation assigned by `gtk_widget_size_allocate()`, since it isn't stored; if a container implementation needs that information it

will have to track it itself.

Parameters

widget	a GtkWidget
allocation	a pointer to a GtkAllocation to copy [out] to.

Since: 2.18

---

gtk_widget_set_allocation ()

```
void
gtk_widget_set_allocation (GtkWidget *widget,
                           const GtkAllocation *allocation);
```

Sets the widget's allocation. This should not be used directly, but from within a widget's size_allocate method.

The allocation set should be the "adjusted" or actual allocation. If you're implementing a GtkContainer, you want to use `gtk_widget_size_allocate()` instead of `gtk_widget_set_allocation()`. The GtkWidgetClass::adjust_size_allocation virtual method adjusts the allocation inside `gtk_widget_size_allocate()` to create an adjusted allocation.

Parameters

widget	a GtkWidget
allocation	a pointer to a GtkAllocation to copy from

Since: 2.18

---

gtk_widget_get_allocated_baseline ()

```
int
gtk_widget_get_allocated_baseline (GtkWidget *widget);
```

Returns the baseline that has currently been allocated to `widget` . This function is intended to be used when implementing handlers for the "draw" function, and when allocating child widgets in "size_allocate".

Parameters

widget	the widget to query

Returns

the baseline of the `widget` , or -1 if none

Since: 3.10

---

gtk_widget_get_allocated_size ()

```
void
gtk_widget_get_allocated_size (GtkWidget *widget,
                               GtkAllocation *allocation,
                               int *baseline);
```

Retrieves the widget's allocated size.

This function returns the last values passed to `gtk_widget_size_allocate_with_baseline()`. The value differs from the size returned in `gtk_widget_get_allocation()` in that functions like `gtk_widget_set_halign()` can adjust the allocation, but not the value returned by this function.

If a widget is not visible, its allocated size is 0.

Parameters

widget	a GtkWidget	
allocation	a pointer to a GtkAllocation to copy to.	[out]
baseline	a pointer to an integer to copy to.	[out][allow-none]

Since: 3.20

---

gtk_widget_get_clip ()

```
void
gtk_widget_get_clip (GtkWidget *widget,
                     GtkAllocation *clip);
```

Retrieves the widget's clip area.

The clip area is the area in which all of `widget` 's drawing will happen. Other toolkits call it the bounding box.

Historically, in GTK+ the clip area has been equal to the allocation retrieved via `gtk_widget_get_allocation()`.

Parameters

widget	a GtkWidget	
clip	a pointer to a GtkAllocation to copy to.	[out]

Since: 3.14

---

gtk_widget_set_clip ()

```
void
gtk_widget_set_clip (GtkWidget *widget,
                     const GtkAllocation *clip);
```

Sets the widget's clip. This must not be used directly, but from within a widget's size_allocate method. It must

be called after `gtk_widget_set_allocation()` (or after chaining up to the parent class), because that function resets the clip.

The clip set should be the area that `widget` draws on. If `widget` is a GtkContainer, the area must contain all children's clips.

If this function is not called by `widget` during a ::size-allocate handler, the clip will be set to `widget` 's allocation.

Parameters

widget	a GtkWidget
clip	a pointer to a GtkAllocation to copy from

Since: 3.14

---

gtk_widget_get_app_paintable ()

```
gboolean
gtk_widget_get_app_paintable (GtkWidget *widget);
```

Determines whether the application intends to draw on the widget in an "draw" handler.

See `gtk_widget_set_app_paintable()`

Parameters

widget	a GtkWidget

Returns

`TRUE` if the widget is app paintable

Since: 2.18

---

gtk_widget_get_can_default ()

```
gboolean
gtk_widget_get_can_default (GtkWidget *widget);
```

Determines whether `widget` can be a default widget. See `gtk_widget_set_can_default()`.

Parameters

widget	a GtkWidget

Returns

`TRUE` if `widget` can be a default widget, `FALSE` otherwise

Since: 2.18

---

gtk_widget_set_can_default ()

```
void
gtk_widget_set_can_default (GtkWidget *widget,
                            gboolean can_default);
```

Specifies whether `widget` can be a default widget. See `gtk_widget_grab_default()` for details about the meaning of “default”.

Parameters

widget	a GtkWidget
can_default	whether or not `widget` can be a default widget.

Since: 2.18

---

gtk_widget_get_can_focus ()

```
gboolean
gtk_widget_get_can_focus (GtkWidget *widget);
```

Determines whether `widget` can own the input focus. See `gtk_widget_set_can_focus()`.

Parameters

widget	a GtkWidget

Returns

`TRUE` if `widget` can own the input focus, `FALSE` otherwise

Since: 2.18

---

gtk_widget_set_can_focus ()

```
void
gtk_widget_set_can_focus (GtkWidget *widget,
                          gboolean can_focus);
```

Specifies whether `widget` can own the input focus. See `gtk_widget_grab_focus()` for actually setting the input focus on a widget.

Parameters

widget	a GtkWidget
can_focus	whether or not `widget` can own the input focus.

Since: 2.18

---

gtk_widget_get_focus_on_click ()

```
gboolean
gtk_widget_get_focus_on_click (GtkWidget *widget);
```

Returns whether the widget should grab focus when it is clicked with the mouse. See `gtk_widget_set_focus_on_click()`.

Parameters

widget	a GtkWidget

Returns

`TRUE` if the widget should grab focus when it is clicked with the mouse.

Since: 3.20

---

gtk_widget_set_focus_on_click ()

```
void
gtk_widget_set_focus_on_click (GtkWidget *widget,
                               gboolean focus_on_click);
```

Sets whether the widget should grab focus when it is clicked with the mouse. Making mouse clicks not grab focus is useful in places like toolbars where you don’t want the keyboard focus removed from the main area of the application.

Parameters

widget	a GtkWidget
focus_on_click	whether the widget should grab focus when clicked with the mouse

Since: 3.20

---

gtk_widget_get_double_buffered ()

```
gboolean
gtk_widget_get_double_buffered (GtkWidget *widget);
```

`gtk_widget_get_double_buffered` is deprecated and should not be used in newly-written code.

Determines whether the widget is double buffered.

See gtk_widget_set_double_buffered()

Parameters

widget	a GtkWidget

Returns

TRUE if the widget is double buffered

Since: 2.18

---

gtk_widget_get_has_window ()

```
gboolean
gtk_widget_get_has_window (GtkWidget *widget);
```

Determines whether widget has a GdkWindow of its own. See gtk_widget_set_has_window().

Parameters

widget	a GtkWidget

Returns

TRUE if widget has a window, FALSE otherwise

Since: 2.18

---

gtk_widget_set_has_window ()

```
void
gtk_widget_set_has_window (GtkWidget *widget,
                           gboolean has_window);
```

Specifies whether widget has a GdkWindow of its own. Note that all realized widgets have a non-NULL "window" pointer (gtk_widget_get_window() never returns a NULL window when a widget is realized), but for many of them it's actually the GdkWindow of one of its parent widgets. Widgets that do not create a window for themselves in "realize" must announce this by calling this function with has_window = FALSE.

This function should only be called by widget implementations, and they should call it in their init() function.

Parameters

widget	a GtkWidget
has_window	whether or not widget has a

window.

Since: 2.18

---

gtk_widget_get_sensitive ()

```
gboolean
gtk_widget_get_sensitive (GtkWidget *widget);
```

Returns the widget's sensitivity (in the sense of returning the value that has been set using `gtk_widget_set_sensitive()`).

The effective sensitivity of a widget is however determined by both its own and its parent widget's sensitivity. See `gtk_widget_is_sensitive()`.

Parameters

widget	a GtkWidget

Returns

`TRUE` if the widget is sensitive

Since: 2.18

---

gtk_widget_is_sensitive ()

```
gboolean
gtk_widget_is_sensitive (GtkWidget *widget);
```

Returns the widget's effective sensitivity, which means it is sensitive itself and also its parent widget is sensitive

Parameters

widget	a GtkWidget

Returns

`TRUE` if the widget is effectively sensitive

Since: 2.18

---

gtk_widget_get_state ()

```
GtkStateType
gtk_widget_get_state (GtkWidget *widget);
```

`gtk_widget_get_state` has been deprecated since version 3.0 and should not be used in newly-written code.

Use `gtk_widget_get_state_flags()` instead.

Returns the widget's state. See `gtk_widget_set_state()`.

Parameters

widget	a GtkWidget

Returns

the state of `widget` .

Since: 2.18

---

gtk_widget_get_visible ()

```
gboolean
gtk_widget_get_visible (GtkWidget *widget);
```

Determines whether the widget is visible. If you want to take into account whether the widget's parent is also marked as visible, use `gtk_widget_is_visible()` instead.

This function does not check if the widget is obscured in any way.

See `gtk_widget_set_visible()`.

Parameters

widget	a GtkWidget

Returns

`TRUE` if the widget is visible

Since: 2.18

---

gtk_widget_is_visible ()

```
gboolean
gtk_widget_is_visible (GtkWidget *widget);
```

Determines whether the widget and all its parents are marked as visible.

This function does not check if the widget is obscured in any way.

See also `gtk_widget_get_visible()` and `gtk_widget_set_visible()`

Parameters

widget	a GtkWidget

Returns

`TRUE` if the widget and all its parents are visible

Since: 3.8

---

gtk_widget_set_visible ()

```
void
gtk_widget_set_visible (GtkWidget *widget,
                        gboolean visible);
```

Sets the visibility state of `widget` . Note that setting this to `TRUE` doesn't mean the widget is actually viewable, see `gtk_widget_get_visible()`.

This function simply calls `gtk_widget_show()` or `gtk_widget_hide()` but is nicer to use when the visibility of the widget depends on some condition.

Parameters

widget	a GtkWidget
visible	whether the widget should be shown or not

Since: 2.18

---

gtk_widget_set_state_flags ()

```
void
gtk_widget_set_state_flags (GtkWidget *widget,
                            GtkStateFlags flags,
                            gboolean clear);
```

This function is for use in widget implementations. Turns on flag values in the current widget state (insensitive, prelighted, etc.).

This function accepts the values `GTK_STATE_FLAG_DIR_LTR` and `GTK_STATE_FLAG_DIR_RTL` but ignores them. If you want to set the widget's direction, use `gtk_widget_set_direction()`.

It is worth mentioning that any other state than `GTK_STATE_FLAG_INSENSITIVE`, will be propagated down to all non-internal children if `widget` is a GtkContainer, while `GTK_STATE_FLAG_INSENSITIVE` itself will be propagated down to all GtkContainer children by different means than turning on the state flag down the hierarchy, both `gtk_widget_get_state_flags()` and `gtk_widget_is_sensitive()` will make use of these.

Parameters

widget	a GtkWidget
flags	State flags to turn on
clear	Whether to clear state before turning on `flags`

Since: 3.0

---

gtk_widget_unset_state_flags ()

```
void
gtk_widget_unset_state_flags (GtkWidget *widget,
                              GtkStateFlags flags);
```

This function is for use in widget implementations. Turns off flag values for the current widget state (insensitive, prelighted, etc.). See `gtk_widget_set_state_flags()`.

Parameters

widget	a GtkWidget
flags	State flags to turn off

Since: 3.0

---

gtk_widget_get_state_flags ()

```
GtkStateFlags
gtk_widget_get_state_flags (GtkWidget *widget);
```

Returns the widget state as a flag set. It is worth mentioning that the effective `GTK_STATE_FLAG_INSENSITIVE` state will be returned, that is, also based on parent insensitivity, even if `widget` itself is sensitive.

Also note that if you are looking for a way to obtain the GtkStateFlags to pass to a GtkStyleContext method, you should look at `gtk_style_context_get_state()`.

Parameters

widget	a GtkWidget

Returns

The state flags for widget

Since: 3.0

---

gtk_widget_has_default ()

```
gboolean
gtk_widget_has_default (GtkWidget *widget);
```

Determines whether `widget` is the current default widget within its toplevel. See `gtk_widget_set_can_default()`.

Parameters

widget	a GtkWidget

Returns

TRUE if widget is the current default widget within its toplevel, FALSE otherwise

Since: 2.18

---

gtk_widget_has_focus ()

```
gboolean
gtk_widget_has_focus (GtkWidget *widget);
```

Determines if the widget has the global input focus. See gtk_widget_is_focus() for the difference between having the global input focus, and only having the focus within a toplevel.

Parameters

widget	a GtkWidget

Returns

TRUE if the widget has the global input focus.

Since: 2.18

---

gtk_widget_has_visible_focus ()

```
gboolean
gtk_widget_has_visible_focus (GtkWidget *widget);
```

Determines if the widget should show a visible indication that it has the global input focus. This is a convenience function for use in ::draw handlers that takes into account whether focus indication should currently be shown in the toplevel window of widget . See gtk_window_get_focus_visible() for more information about focus indication.

To find out if the widget has the global input focus, use gtk_widget_has_focus().

Parameters

widget	a GtkWidget

Returns

TRUE if the widget should display a “focus rectangle”

Since: 3.2

---

gtk_widget_has_grab ()

```
gboolean
gtk_widget_has_grab (GtkWidget *widget);
```

Determines whether the widget is currently grabbing events, so it is the only widget receiving input events (keyboard and mouse).

See also `gtk_grab_add()`.

Parameters

widget	a GtkWidget

Returns

`TRUE` if the widget is in the grab_widgets stack

Since: 2.18

---

gtk_widget_has_rc_style ()

```
gboolean
gtk_widget_has_rc_style (GtkWidget *widget);
```

`gtk_widget_has_rc_style` has been deprecated since version 3.0 and should not be used in newly-written code.

Use GtkStyleContext instead

Determines if the widget style has been looked up through the rc mechanism.

Parameters

widget	a GtkWidget

Returns

`TRUE` if the widget has been looked up through the rc mechanism, `FALSE` otherwise.

Since: 2.20

---

gtk_widget_is_drawable ()

```
gboolean
gtk_widget_is_drawable (GtkWidget *widget);
```

Determines whether `widget` can be drawn to. A widget can be drawn to if it is mapped and visible.

Parameters

widget	a GtkWidget	

Returns

`TRUE` if `widget` is drawable, `FALSE` otherwise

Since: 2.18

---

gtk_widget_is_toplevel ()

```
gboolean
gtk_widget_is_toplevel (GtkWidget *widget);
```

Determines whether `widget` is a toplevel widget.

Currently only GtkWindow and GtkInvisible (and out-of-process GtkPlugs) are toplevel widgets. Toplevel widgets have no parent widget.

Parameters

widget	a GtkWidget	

Returns

`TRUE` if `widget` is a toplevel, `FALSE` otherwise

Since: 2.18

---

gtk_widget_set_window ()

```
void
gtk_widget_set_window (GtkWidget *widget,
                       GdkWindow *window);
```

Sets a widget's window. This function should only be used in a widget's "realize" implementation. The `window` passed is usually either new window created with `gdk_window_new()`, or the window of its parent widget as returned by `gtk_widget_get_parent_window()`.

Widgets must indicate whether they will create their own GdkWindow by calling `gtk_widget_set_has_window()`. This is usually done in the widget's `init()` function.

Note that this function does not add any reference to `window` .

Parameters

widget	a GtkWidget	
window	a GdkWindow.	[transfer full]

Since: 2.18

---

gtk_widget_set_receives_default ()

```
void
gtk_widget_set_receives_default (GtkWidget *widget,
                                 gboolean receives_default);
```

Specifies whether `widget` will be treated as the default widget within its toplevel when it has the focus, even if another widget is the default.

See `gtk_widget_grab_default()` for details about the meaning of “default”.

Parameters

widget	a GtkWidget
receives_default	whether or not `widget` can be a default widget.

Since: 2.18

---

gtk_widget_get_receives_default ()

```
gboolean
gtk_widget_get_receives_default (GtkWidget *widget);
```

Determines whether `widget` is always treated as the default widget within its toplevel when it has the focus, even if another widget is the default.

See `gtk_widget_set_receives_default()`.

Parameters

widget	a GtkWidget

Returns

`TRUE` if `widget` acts as the default widget when focused, `FALSE` otherwise

Since: 2.18

---

gtk_widget_set_support_multidevice ()

```
void
gtk_widget_set_support_multidevice (GtkWidget *widget,
                                    gboolean support_multidevice);
```

Enables or disables multiple pointer awareness. If this setting is `TRUE`, `widget` will start receiving multiple, per device enter/leave events. Note that if custom GdkWindows are created in “realize”, `gdk_window_set_support_multidevice()` will have to be called manually on them.

Parameters

widget	a GtkWidget
support_multidevice	TRUE to support input from multiple devices.

Since: 3.0

---

gtk_widget_get_support_multidevice ()

```
gboolean
gtk_widget_get_support_multidevice (GtkWidget *widget);
```

Returns TRUE if widget is multiple pointer aware. See gtk_widget_set_support_multidevice() for more information.

Parameters

widget	a GtkWidget

Returns

TRUE if widget is multidevice aware.

---

gtk_widget_set_realized ()

```
void
gtk_widget_set_realized (GtkWidget *widget,
                         gboolean realized);
```

Marks the widget as being realized. This function must only be called after all GdkWindows for the widget have been created and registered.

This function should only ever be called in a derived widget's “realize” or “unrealize” implementation.

Parameters

widget	a GtkWidget
realized	TRUE to mark the widget as realized

Since: 2.20

---

gtk_widget_get_realized ()

```
gboolean
gtk_widget_get_realized (GtkWidget *widget);
```

Determines whether widget is realized.

Parameters

widget	a GtkWidget

Returns

TRUE if widget is realized, FALSE otherwise

Since: 2.20

---

gtk_widget_set_mapped ()

```
void
gtk_widget_set_mapped (GtkWidget *widget,
                       gboolean mapped);
```

Marks the widget as being mapped.

This function should only ever be called in a derived widget's “map” or “unmap” implementation.

Parameters

widget	a GtkWidget
mapped	TRUE to mark the widget as mapped

Since: 2.20

---

gtk_widget_get_mapped ()

```
gboolean
gtk_widget_get_mapped (GtkWidget *widget);
```

Whether the widget is mapped.

Parameters

widget	a GtkWidget

Returns

TRUE if the widget is mapped, FALSE otherwise.

Since: 2.20

---

gtk_widget_get_requisition ()

```
void
gtk_widget_get_requisition (GtkWidget *widget,
                            GtkRequisition *requisition);
```

`gtk_widget_get_requisition` has been deprecated since version 3.0 and should not be used in newly-written code.

The GtkRequisition cache on the widget was removed, If you need to cache sizes across requests and allocations, add an explicit cache to the widget in question instead.

Retrieves the widget's requisition.

This function should only be used by widget implementations in order to figure whether the widget's requisition has actually changed after some internal state change (so that they can call `gtk_widget_queue_resize()` instead of `gtk_widget_queue_draw()`).

Normally, `gtk_widget_size_request()` should be used.

Parameters

widget	a GtkWidget	
requisition	a pointer to a GtkRequisition to copy to.	[out]

Since: 2.20

---

gtk_widget_device_is_shadowed ()

```
gboolean
gtk_widget_device_is_shadowed (GtkWidget *widget,
                               GdkDevice *device);
```

Returns `TRUE` if `device` has been shadowed by a GTK+ device grab on another widget, so it would stop sending events to `widget` . This may be used in the "grab-notify" signal to check for specific devices. See `gtk_device_grab_add()`.

Parameters

widget	a GtkWidget
device	a GdkDevice

Returns

`TRUE` if there is an ongoing grab on `device` by another GtkWidget than `widget` .

Since: 3.0

---

gtk_widget_get_modifier_mask ()

```
GdkModifierType
gtk_widget_get_modifier_mask (GtkWidget *widget,
                              GdkModifierIntent intent);
```

Returns the modifier mask the `widget` 's windowing system backend uses for a particular purpose.

See `gdk_keymap_get_modifier_mask()`.

Parameters

widget	a GtkWidget
intent	the use case for the modifier mask

Returns

the modifier mask used for `intent`.

Since: 3.4

---

gtk_widget_insert_action_group ()

```
void
gtk_widget_insert_action_group (GtkWidget *widget,
                                const gchar *name,
                                GActionGroup *group);
```

Inserts `group` into `widget`. Children of `widget` that implement GtkActionable can then be associated with actions in `group` by setting their "action-name" to `prefix`.`action-name`.

If `group` is `NULL`, a previously inserted group for `name` is removed from `widget`.

Parameters

widget	a GtkWidget	
name	the prefix for actions in `group`	
group	a GActionGroup, or `NULL`.	[allow-none]

Since: 3.6

---

gtk_widget_get_opacity ()

```
double
gtk_widget_get_opacity (GtkWidget *widget);
```

Fetches the requested opacity for this widget. See `gtk_widget_set_opacity()`.

Parameters

widget	a GtkWidget

Returns

the requested opacity for this widget.

Since: 3.8

---

gtk_widget_set_opacity ()

```
void
gtk_widget_set_opacity (GtkWidget *widget,
                        double opacity);
```

Request the `widget` to be rendered partially transparent, with opacity 0 being fully transparent and 1 fully opaque. (Opacity values are clamped to the [0,1] range.). This works on both toplevel widget, and child widgets, although there are some limitations:

For toplevel widgets this depends on the capabilities of the windowing system. On X11 this has any effect only on X screens with a compositing manager running. See `gtk_widget_is_composited()`. On Windows it should work always, although setting a window's opacity after the window has been shown causes it to flicker once on Windows.

For child widgets it doesn't work if any affected widget has a native window, or disables double buffering.

Parameters

widget	a GtkWidget
opacity	desired opacity, between 0 and 1

Since: 3.8

---

gtk_widget_list_action_prefixes ()

```
const gchar **
gtk_widget_list_action_prefixes (GtkWidget *widget);
```

Retrieves a `NULL`-terminated array of strings containing the prefixes of GActionGroup's available to `widget` .

Parameters

widget	A GtkWidget

Returns

a `NULL`-terminated array of strings.

[transfer container]

Since: 3.16

---

gtk_widget_get_action_group ()

```
GActionGroup *
gtk_widget_get_action_group (GtkWidget *widget,
                             const gchar *prefix);
```

Retrieves the GActionGroup that was registered using `prefix` . The resulting GActionGroup may have been registered to `widget` or any GtkWidget in its ancestry.

If no action group was found matching `prefix` , then `NULL` is returned.

Parameters

widget	A GtkWidget
prefix	The “prefix” of the action group.

Returns

A GActionGroup or `NULL`.

[transfer none][nullable]

Since: 3.16

---

gtk_widget_get_path ()

```
GtkWidgetPath *
gtk_widget_get_path (GtkWidget *widget);
```

Returns the GtkWidgetPath representing `widget` , if the widget is not connected to a toplevel widget, a partial path will be created.

Parameters

widget	a GtkWidget

Returns

The GtkWidgetPath representing `widget` .

[transfer none]

---

gtk_widget_get_style_context ()

```
GtkStyleContext *
gtk_widget_get_style_context (GtkWidget *widget);
```

Returns the style context associated to `widget` . The returned object is guaranteed to be the same for the lifetime of `widget` .

Parameters

widget	a GtkWidget

Returns

a GtkStyleContext. This memory is owned by `widget` and must not be freed.

[transfer none]

---

gtk_widget_reset_style ()

```
void
gtk_widget_reset_style (GtkWidget *widget);
```

Updates the style context of `widget` and all descendants by updating its widget path. GtkContainers may want to use this on a child when reordering it in a way that a different style might apply to it. See also `gtk_container_get_path_for_child()`.

Parameters

widget	a GtkWidget

Since: 3.0

---

gtk_widget_class_get_css_name ()

```
const char *
gtk_widget_class_get_css_name (GtkWidgetClass *widget_class);
```

Gets the name used by this class for matching in CSS code. See `gtk_widget_class_set_css_name()` for details.

Parameters

widget_class	class to set the name on

Returns

the CSS name of the given class

Since: 3.20

---

gtk_widget_class_set_css_name ()

```
void
gtk_widget_class_set_css_name (GtkWidgetClass *widget_class,
                               const char *name);
```

Sets the name to be used for CSS matching of widgets.

If this function is not called for a given class, the name of the parent class is used.

Parameters

widget_class	class to set the name on
name	name to use

Since: 3.20

---

gtk_requisition_new ()

```
GtkRequisition *
gtk_requisition_new (void);
```

Allocates a new GtkRequisition and initializes its elements to zero.

Returns

a new empty GtkRequisition. The newly allocated GtkRequisition should be freed with `gtk_requisition_free()`.

Since: 3.0

---

gtk_requisition_copy ()

```
GtkRequisition *
gtk_requisition_copy (const GtkRequisition *requisition);
```

Copies a GtkRequisition.

Parameters

requisition	a GtkRequisition

Returns

a copy of `requisition`

---

gtk_requisition_free ()

```
void
gtk_requisition_free (GtkRequisition *requisition);
```

Frees a GtkRequisition.

Parameters

requisition	a GtkRequisition

---

gtk_widget_get_preferred_height ()

```
void
gtk_widget_get_preferred_height (GtkWidget *widget,
                                 gint *minimum_height,
                                 gint *natural_height);
```

Retrieves a widget’s initial minimum and natural height.

This call is specific to width-for-height requests.

The returned request will be modified by the GtkWidgetClass::adjust_size_request virtual method and by any GtkSizeGroups that have been applied. That is, the returned request is the one that should be used for layout, not necessarily the one returned by the widget itself.

Parameters

widget	a GtkWidget instance	
minimum_height	location to store the minimum height, or `NULL`.	[out][allow-none]
natural_height	location to store the natural height, or `NULL`.	[out][allow-none]

Since: 3.0

---

gtk_widget_get_preferred_width ()

```
void
gtk_widget_get_preferred_width (GtkWidget *widget,
                                gint *minimum_width,
                                gint *natural_width);
```

Retrieves a widget’s initial minimum and natural width.

This call is specific to height-for-width requests.

The returned request will be modified by the GtkWidgetClass::adjust_size_request virtual method and by any GtkSizeGroups that have been applied. That is, the returned request is the one that should be used for layout, not necessarily the one returned by the widget itself.

Parameters

widget	a GtkWidget instance	
minimum_width	location to store the minimum width, or `NULL`.	[out][allow-none]
natural_width	location to store the natural width, or `NULL`.	[out][allow-none]

Since: 3.0

---

gtk_widget_get_preferred_height_for_width ()

```
void
```

```
gtk_widget_get_preferred_height_for_width
                                (GtkWidget *widget,
                                 gint width,
                                 gint *minimum_height,
                                 gint *natural_height);
```

Retrieves a widget’s minimum and natural height if it would be given the specified `width` .

The returned request will be modified by the GtkWidgetClass::adjust_size_request virtual method and by any GtkSizeGroups that have been applied. That is, the returned request is the one that should be used for layout, not necessarily the one returned by the widget itself.

Parameters

widget	a GtkWidget instance	
width	the width which is available for allocation	
minimum_height	location for storing the minimum height, or `NULL`.	[out][allow-none]
natural_height	location for storing the natural height, or `NULL`.	[out][allow-none]

Since: 3.0

---

gtk_widget_get_preferred_width_for_height ()

```
void
gtk_widget_get_preferred_width_for_height
                                (GtkWidget *widget,
                                 gint height,
                                 gint *minimum_width,
                                 gint *natural_width);
```

Retrieves a widget’s minimum and natural width if it would be given the specified `height` .

The returned request will be modified by the GtkWidgetClass::adjust_size_request virtual method and by any GtkSizeGroups that have been applied. That is, the returned request is the one that should be used for layout, not necessarily the one returned by the widget itself.

Parameters

widget	a GtkWidget instance	
height	the height which is available for allocation	
minimum_width	location for storing the minimum width, or `NULL`.	[out][allow-none]
natural_width	location for storing the natural width, or `NULL`.	[out][allow-none]

Since: 3.0

---

gtk_widget_get_preferred_height_and_baseline_for_width ()

```
void
gtk_widget_get_preferred_height_and_baseline_for_width
                                (GtkWidget *widget,
                                 gint width,
                                 gint *minimum_height,
                                 gint *natural_height,
                                 gint *minimum_baseline,
                                 gint *natural_baseline);
```

Retrieves a widget's minimum and natural height and the corresponding baselines if it would be given the specified `width` , or the default height if `width` is -1. The baselines may be -1 which means that no baseline is requested for this widget.

The returned request will be modified by the GtkWidgetClass::adjust_size_request and GtkWidgetClass::adjust_baseline_request virtual methods and by any GtkSizeGroups that have been applied. That is, the returned request is the one that should be used for layout, not necessarily the one returned by the widget itself.

Parameters

widget	a GtkWidget instance	
width	the width which is available for allocation, or -1 if none	
minimum_height	location for storing the minimum height, or `NULL`.	[out][allow-none]
natural_height	location for storing the natural height, or `NULL`.	[out][allow-none]
minimum_baseline	location for storing the baseline for the minimum height, or `NULL`.	[out][allow-none]
natural_baseline	location for storing the baseline for the natural height, or `NULL`.	[out][allow-none]

Since: 3.10

---

gtk_widget_get_request_mode ()

```
GtkSizeRequestMode
gtk_widget_get_request_mode (GtkWidget *widget);
```

Gets whether the widget prefers a height-for-width layout or a width-for-height layout.

GtkBin widgets generally propagate the preference of their child, container widgets need to request something either in context of their children or in context of their allocation capabilities.

Parameters

widget	a GtkWidget instance

Returns

The GtkSizeRequestMode preferred by `widget` .

Since: 3.0

---

gtk_widget_get_preferred_size ()

```
void
gtk_widget_get_preferred_size (GtkWidget *widget,
                               GtkRequisition *minimum_size,
                               GtkRequisition *natural_size);
```

Retrieves the minimum and natural size of a widget, taking into account the widget's preference for height-for-width management.

This is used to retrieve a suitable size by container widgets which do not impose any restrictions on the child placement. It can be used to deduce toplevel window and menu sizes as well as child widgets in free-form containers such as GtkLayout.

Handle with care. Note that the natural height of a height-for-width widget will generally be a smaller size than the minimum height, since the required height for the natural width is generally smaller than the required height for the minimum width.

Use `gtk_widget_get_preferred_height_and_baseline_for_width()` if you want to support baseline alignment.

Parameters

widget	a GtkWidget instance	
minimum_size	location for storing the minimum size, or `NULL`.	[out][allow-none]
natural_size	location for storing the natural size, or `NULL`.	[out][allow-none]

Since: 3.0

---

gtk_distribute_natural_allocation ()

```
gint
gtk_distribute_natural_allocation (gint extra_space,
                                   guint n_requested_sizes,
                                   GtkRequestedSize *sizes);
```

Distributes `extra_space` to child `sizes` by bringing smaller children up to natural size first.

The remaining space will be added to the `minimum_size` member of the GtkRequestedSize struct. If all sizes reach their natural size then the remaining space is returned.

Parameters

extra_space	Extra space to redistribute among children after subtracting minimum sizes and any child padding from the overall allocation

n_requested_sizes	Number of requests to fit into the allocation
sizes	An array of structs with a client pointer and a minimum/natural size in the orientation of the allocation.

Returns

The remainder of `extra_space` after redistributing space to `sizes` .

---

gtk_widget_get_halign ()

```
GtkAlign
gtk_widget_get_halign (GtkWidget *widget);
```

Gets the value of the “halign” property.

For backwards compatibility reasons this method will never return `GTK_ALIGN_BASELINE`, but instead it will convert it to `GTK_ALIGN_FILL`. Baselines are not supported for horizontal alignment.

Parameters

widget	a GtkWidget

Returns

the horizontal alignment of `widget`

---

gtk_widget_set_halign ()

```
void
gtk_widget_set_halign (GtkWidget *widget,
                       GtkAlign align);
```

Sets the horizontal alignment of `widget` . See the “halign” property.

Parameters

widget	a GtkWidget
align	the horizontal alignment

---

gtk_widget_get_valign ()

```
GtkAlign
gtk_widget_get_valign (GtkWidget *widget);
```

Gets the value of the “valign” property.

For backwards compatibility reasons this method will never return GTK_ALIGN_BASELINE, but instead it will convert it to GTK_ALIGN_FILL. If your widget want to support baseline aligned children it must use gtk_widget_get_valign_with_baseline(), or g_object_get (widget, "valign", &value, NULL), which will also report the true value.

Parameters

widget	a GtkWidget

Returns

the vertical alignment of widget , ignoring baseline alignment

---

gtk_widget_get_valign_with_baseline ()

```
GtkAlign
gtk_widget_get_valign_with_baseline (GtkWidget *widget);
```

Gets the value of the "valign" property, including GTK_ALIGN_BASELINE.

Parameters

widget	a GtkWidget

Returns

the vertical alignment of widget

Since: 3.10

---

gtk_widget_set_valign ()

```
void
gtk_widget_set_valign (GtkWidget *widget,
                       GtkAlign align);
```

Sets the vertical alignment of widget . See the "valign" property.

Parameters

widget	a GtkWidget
align	the vertical alignment

---

gtk_widget_get_margin_left ()

```
gint
gtk_widget_get_margin_left (GtkWidget *widget);
```

`gtk_widget_get_margin_left` has been deprecated since version 3.12 and should not be used in newly-written code.

Use `gtk_widget_get_margin_start()` instead.

Gets the value of the "margin-left" property.

Parameters

widget	a GtkWidget

Returns

The left margin of `widget`

Since: 3.0

---

gtk_widget_set_margin_left ()

```
void
gtk_widget_set_margin_left (GtkWidget *widget,
                            gint margin);
```

`gtk_widget_set_margin_left` has been deprecated since version 3.12 and should not be used in newly-written code.

Use `gtk_widget_set_margin_start()` instead.

Sets the left margin of `widget` . See the "margin-left" property.

Parameters

widget	a GtkWidget
margin	the left margin

Since: 3.0

---

gtk_widget_get_margin_right ()

```
gint
gtk_widget_get_margin_right (GtkWidget *widget);
```

`gtk_widget_get_margin_right` has been deprecated since version 3.12 and should not be used in newly-written code.

Use `gtk_widget_get_margin_end()` instead.

Gets the value of the "margin-right" property.

Parameters

widget	a GtkWidget

Returns

The right margin of `widget`

Since: 3.0

---

gtk_widget_set_margin_right ()

```
void
gtk_widget_set_margin_right (GtkWidget *widget,
                             gint margin);
```

`gtk_widget_set_margin_right` has been deprecated since version 3.12 and should not be used in newly-written code.

Use `gtk_widget_set_margin_end()` instead.

Sets the right margin of `widget` . See the "margin-right" property.

Parameters

widget	a GtkWidget
margin	the right margin

Since: 3.0

---

gtk_widget_get_margin_start ()

```
gint
gtk_widget_get_margin_start (GtkWidget *widget);
```

Gets the value of the "margin-start" property.

Parameters

widget	a GtkWidget

Returns

The start margin of `widget`

Since: 3.12

---

gtk_widget_set_margin_start ()

```
void
gtk_widget_set_margin_start (GtkWidget *widget,
                             gint margin);
```

Sets the start margin of `widget` . See the “margin-start” property.

Parameters

widget	a GtkWidget
margin	the start margin

Since: 3.12

---

gtk_widget_get_margin_end ()

```
gint
gtk_widget_get_margin_end (GtkWidget *widget);
```

Gets the value of the “margin-end” property.

Parameters

widget	a GtkWidget

Returns

The end margin of `widget`

Since: 3.12

---

gtk_widget_set_margin_end ()

```
void
gtk_widget_set_margin_end (GtkWidget *widget,
                           gint margin);
```

Sets the end margin of `widget` . See the “margin-end” property.

Parameters

widget	a GtkWidget
margin	the end margin

Since: 3.12

---

gtk_widget_get_margin_top ()

```
gint
```

```
gtk_widget_get_margin_top (GtkWidget *widget);
```
Gets the value of the “margin-top” property.

Parameters

widget	a GtkWidget

Returns

The top margin of `widget`

Since: 3.0

---

gtk_widget_set_margin_top ()

```
void
gtk_widget_set_margin_top (GtkWidget *widget,
                           gint margin);
```
Sets the top margin of `widget` . See the “margin-top” property.

Parameters

widget	a GtkWidget
margin	the top margin

Since: 3.0

---

gtk_widget_get_margin_bottom ()

```
gint
gtk_widget_get_margin_bottom (GtkWidget *widget);
```
Gets the value of the “margin-bottom” property.

Parameters

widget	a GtkWidget

Returns

The bottom margin of `widget`

Since: 3.0

---

gtk_widget_set_margin_bottom ()

```
void
gtk_widget_set_margin_bottom (GtkWidget *widget,
                              gint margin);
```

Sets the bottom margin of `widget` . See the “margin-bottom” property.

Parameters

widget	a GtkWidget
margin	the bottom margin

Since: 3.0

---

gtk_widget_get_hexpand ()

```
gboolean
gtk_widget_get_hexpand (GtkWidget *widget);
```

Gets whether the widget would like any available extra horizontal space. When a user resizes a GtkWindow, widgets with expand=TRUE generally receive the extra space. For example, a list or scrollable area or document in your window would often be set to expand.

Containers should use `gtk_widget_compute_expand()` rather than this function, to see whether a widget, or any of its children, has the expand flag set. If any child of a widget wants to expand, the parent may ask to expand also.

This function only looks at the widget’s own hexpand flag, rather than computing whether the entire widget tree rooted at this widget wants to expand.

Parameters

widget	the widget

Returns

whether hexpand flag is set

---

gtk_widget_set_hexpand ()

```
void
gtk_widget_set_hexpand (GtkWidget *widget,
                        gboolean expand);
```

Sets whether the widget would like any available extra horizontal space. When a user resizes a GtkWindow, widgets with expand=TRUE generally receive the extra space. For example, a list or scrollable area or document in your window would often be set to expand.

Call this function to set the expand flag if you would like your widget to become larger horizontally when the window has extra room.

By default, widgets automatically expand if any of their children want to expand. (To see if a widget will

automatically expand given its current children and state, call gtk_widget_compute_expand(). A container can decide how the expandability of children affects the expansion of the container by overriding the compute_expand virtual method on GtkWidget.).

Setting hexpand explicitly with this function will override the automatic expand behavior.

This function forces the widget to expand or not to expand, regardless of children. The override occurs because gtk_widget_set_hexpand() sets the hexpand-set property (see gtk_widget_set_hexpand_set()) which causes the widget's hexpand value to be used, rather than looking at children and widget state.

Parameters

widget	the widget
expand	whether to expand

---

gtk_widget_get_hexpand_set ()

```
gboolean
gtk_widget_get_hexpand_set (GtkWidget *widget);
```

Gets whether gtk_widget_set_hexpand() has been used to explicitly set the expand flag on this widget.

If hexpand is set, then it overrides any computed expand value based on child widgets. If hexpand is not set, then the expand value depends on whether any children of the widget would like to expand.

There are few reasons to use this function, but it's here for completeness and consistency.

Parameters

widget	the widget

Returns

whether hexpand has been explicitly set

---

gtk_widget_set_hexpand_set ()

```
void
gtk_widget_set_hexpand_set (GtkWidget *widget,
                            gboolean set);
```

Sets whether the hexpand flag (see gtk_widget_get_hexpand()) will be used.

The hexpand-set property will be set automatically when you call gtk_widget_set_hexpand() to set hexpand, so the most likely reason to use this function would be to unset an explicit expand flag.

If hexpand is set, then it overrides any computed expand value based on child widgets. If hexpand is not set, then the expand value depends on whether any children of the widget would like to expand.

There are few reasons to use this function, but it's here for completeness and consistency.

Parameters

widget	the widget
set	value for hexpand-set property

---

gtk_widget_get_vexpand ()

```
gboolean
gtk_widget_get_vexpand (GtkWidget *widget);
```

Gets whether the widget would like any available extra vertical space.

See `gtk_widget_get_hexpand()` for more detail.

Parameters

widget	the widget

Returns

whether vexpand flag is set

---

gtk_widget_set_vexpand ()

```
void
gtk_widget_set_vexpand (GtkWidget *widget,
                        gboolean expand);
```

Sets whether the widget would like any available extra vertical space.

See `gtk_widget_set_hexpand()` for more detail.

Parameters

widget	the widget
expand	whether to expand

---

gtk_widget_get_vexpand_set ()

```
gboolean
gtk_widget_get_vexpand_set (GtkWidget *widget);
```

Gets whether `gtk_widget_set_vexpand()` has been used to explicitly set the expand flag on this widget.

See `gtk_widget_get_hexpand_set()` for more detail.

Parameters

widget	the widget

Returns

whether vexpand has been explicitly set

---

gtk_widget_set_vexpand_set ()

```
void
gtk_widget_set_vexpand_set (GtkWidget *widget,
                            gboolean set);
```

Sets whether the vexpand flag (see `gtk_widget_get_vexpand()`) will be used.

See `gtk_widget_set_hexpand_set()` for more detail.

Parameters

widget	the widget
set	value for vexpand-set property

---

gtk_widget_queue_compute_expand ()

```
void
gtk_widget_queue_compute_expand (GtkWidget *widget);
```

Mark `widget` as needing to recompute its expand flags. Call this function when setting legacy expand child properties on the child of a container.

See `gtk_widget_compute_expand()`.

Parameters

widget	a GtkWidget

---

gtk_widget_compute_expand ()

```
gboolean
gtk_widget_compute_expand (GtkWidget *widget,
                           GtkOrientation orientation);
```

Computes whether a container should give this widget extra space when possible. Containers should check this, rather than looking at `gtk_widget_get_hexpand()` or `gtk_widget_get_vexpand()`.

This function already checks whether the widget is visible, so visibility does not need to be checked separately. Non-visible widgets are not expanded.

The computed expand value uses either the expand setting explicitly set on the widget itself, or, if none has been

explicitly set, the widget may expand if some of its children do.

Parameters

widget	the widget
orientation	expand direction

Returns

whether widget tree rooted here should be expanded

---

gtk_widget_init_template ()

```
void
gtk_widget_init_template (GtkWidget *widget);
```

Creates and initializes child widgets defined in templates. This function must be called in the instance initializer for any class which assigned itself a template using `gtk_widget_class_set_template()`

It is important to call this function in the instance initializer of a GtkWidget subclass and not in `GObject.constructed()` or `GObject.constructor()` for two reasons.

One reason is that generally derived widgets will assume that parent class composite widgets have been created in their instance initializers.

Another reason is that when calling `g_object_new()` on a widget with composite templates, it's important to build the composite widgets before the construct properties are set. Properties passed to `g_object_new()` should take precedence over properties set in the private template XML.

Parameters

widget	a GtkWidget

Since: 3.10

---

gtk_widget_class_set_template ()

```
void
gtk_widget_class_set_template (GtkWidgetClass *widget_class,
                               GBytes *template_bytes);
```

This should be called at class initialization time to specify the GtkBuilder XML to be used to extend a widget.

For convenience, `gtk_widget_class_set_template_from_resource()` is also provided.

Note that any class that installs templates must call `gtk_widget_init_template()` in the widget's instance initializer.

Parameters

widget_class	A GtkWidgetClass
template_bytes	A GBytes holding the GtkBuilder XML

Since: 3.10

---

gtk_widget_class_set_template_from_resource ()

```
void
gtk_widget_class_set_template_from_resource
                                (GtkWidgetClass *widget_class,
                                 const gchar *resource_name);
```

A convenience function to call `gtk_widget_class_set_template()`.

Note that any class that installs templates must call `gtk_widget_init_template()` in the widget's instance initializer.

Parameters

widget_class	A GtkWidgetClass
resource_name	The name of the resource to load the template from

Since: 3.10

---

gtk_widget_get_template_child ()

```
GObject *
gtk_widget_get_template_child (GtkWidget *widget,
                               GType widget_type,
                               const gchar *name);
```

Fetch an object build from the template XML for `widget_type` in this `widget` instance.

This will only report children which were previously declared with `gtk_widget_class_bind_template_child_full()` or one of its variants.

This function is only meant to be called for code which is private to the `widget_type` which declared the child and is meant for language bindings which cannot easily make use of the GObject structure offsets.

Parameters

widget	A GtkWidget
widget_type	The GType to get a template child for
name	The "id" of the child defined in the template XML

Returns

The object built in the template XML with the id `name` .

[transfer none]

---

gtk_widget_class_bind_template_child()

```
#define gtk_widget_class_bind_template_child(widget_class, TypeName, member_name)
```

Binds a child widget defined in a template to the `widget_class` .

This macro is a convenience wrapper around the `gtk_widget_class_bind_template_child_full()` function.

This macro will use the offset of the `member_name` inside the `TypeName` instance structure.

Parameters

widget_class	a GtkWidgetClass
TypeName	the type name of this widget
member_name	name of the instance member in the instance struct for `data_type`

Since: 3.10

---

gtk_widget_class_bind_template_child_internal()

```
#define gtk_widget_class_bind_template_child_internal(widget_class, TypeName, member_name)
```

Binds a child widget defined in a template to the `widget_class` , and also makes it available as an internal child in GtkBuilder, under the name `member_name` .

This macro is a convenience wrapper around the `gtk_widget_class_bind_template_child_full()` function.

This macro will use the offset of the `member_name` inside the `TypeName` instance structure.

Parameters

widget_class	a GtkWidgetClass
TypeName	the type name, in CamelCase
member_name	name of the instance member in the instance struct for `data_type`

Since: 3.10

---

gtk_widget_class_bind_template_child_private()

```
#define gtk_widget_class_bind_template_child_private(widget_class, TypeName, member_name)
```

Binds a child widget defined in a template to the `widget_class` .

This macro is a convenience wrapper around the `gtk_widget_class_bind_template_child_full()` function.

This macro will use the offset of the `member_name` inside the `TypeName` private data structure (it uses `G_PRIVATE_OFFSET()`, so the private struct must be added with `G_ADD_PRIVATE()`).

Parameters

widget_class	a GtkWidgetClass
TypeName	the type name of this widget
member_name	name of the instance private member in the private struct for `data_type`

Since: 3.10

---

gtk_widget_class_bind_template_child_internal_private()

```
#define             gtk_widget_class_bind_template_child_internal_private(widget_class,
TypeName, member_name)
```

Binds a child widget defined in a template to the `widget_class` , and also makes it available as an internal child in GtkBuilder, under the name `member_name` .

This macro is a convenience wrapper around the `gtk_widget_class_bind_template_child_full()` function.

This macro will use the offset of the `member_name` inside the `TypeName` private data structure.

Parameters

widget_class	a GtkWidgetClass
TypeName	the type name, in CamelCase
member_name	name of the instance private member on the private struct for `data_type`

Since: 3.10

---

gtk_widget_class_bind_template_child_full ()

```
void
gtk_widget_class_bind_template_child_full
                               (GtkWidgetClass *widget_class,
                                const gchar *name,
                                gboolean internal_child,
                                gssize struct_offset);
```

Automatically assign an object declared in the class template XML to be set to a location on a freshly built instance’s private data, or alternatively accessible via `gtk_widget_get_template_child()`.

The struct can point either into the public instance, then you should use G_STRUCT_OFFSET(WidgetType, member) for `struct_offset` , or in the private struct, then you should use G_PRIVATE_OFFSET(WidgetType,

member).

An explicit strong reference will be held automatically for the duration of your instance's life cycle, it will be released automatically when `GObjectClass.dispose()` runs on your instance and if a `struct_offset` that is != 0 is specified, then the automatic location in your instance public or private data will be set to `NULL`. You can however access an automated child pointer the first time your classes `GObjectClass.dispose()` runs, or alternatively in `GtkWidgetClass.destroy()`.

If `internal_child` is specified, `GtkBuildableIface.get_internal_child()` will be automatically implemented by the GtkWidget class so there is no need to implement it manually.

The wrapper macros `gtk_widget_class_bind_template_child()`, `gtk_widget_class_bind_template_child_internal()`, `gtk_widget_class_bind_template_child_private()` and `gtk_widget_class_bind_template_child_internal_private()` might be more convenient to use.

Note that this must be called from a composite widget classes class initializer after calling `gtk_widget_class_set_template()`.

Parameters

widget_class	A GtkWidgetClass
name	The "id" of the child defined in the template XML
internal_child	Whether the child should be accessible as an "internal-child" when this class is used in GtkBuilder XML
struct_offset	The structure offset into the composite widget's instance public or private structure where the automated child pointer should be set, or 0 to not assign the pointer.

Since: 3.10

---

gtk_widget_class_bind_template_callback()

```
#define gtk_widget_class_bind_template_callback(widget_class, callback)
```

Binds a callback function defined in a template to the `widget_class` .

This macro is a convenience wrapper around the `gtk_widget_class_bind_template_callback_full()` function.

Parameters

widget_class	a GtkWidgetClass
callback	the callback symbol

Since: 3.10

---

gtk_widget_class_bind_template_callback_full ()

```
void
gtk_widget_class_bind_template_callback_full
                                (GtkWidgetClass *widget_class,
                                 const gchar *callback_name,
                                 GCallback callback_symbol);
```

Declares a `callback_symbol` to handle `callback_name` from the template XML defined for `widget_type` . See `gtk_builder_add_callback_symbol()`.

Note that this must be called from a composite widget classes class initializer after calling `gtk_widget_class_set_template()`.

Parameters

widget_class	A GtkWidgetClass	
callback_name	The name of the callback as expected in the template XML	
callback_symbol	The callback symbol.	[scope async]

Since: 3.10

---

gtk_widget_class_set_connect_func ()

```
void
gtk_widget_class_set_connect_func (GtkWidgetClass *widget_class,
                                   GtkBuilderConnectFunc connect_func,
                                   gpointer connect_data,
                                   GDestroyNotify connect_data_destroy);
```

For use in language bindings, this will override the default GtkBuilderConnectFunc to be used when parsing GtkBuilder XML from this class's template data.

Note that this must be called from a composite widget classes class initializer after calling `gtk_widget_class_set_template()`.

Parameters

widget_class	A GtkWidgetClass
connect_func	The GtkBuilderConnectFunc to use when connecting signals in the class template
connect_data	The data to pass to `connect_func`
connect_data_destroy	The GDestroyNotify to free `connect_data` , this will only be used at class finalization time, when no classes of type `widget_type` are in use anymore.

Since: 3.10

Types and Values

GtkWidget

```
typedef struct _GtkWidget GtkWidget;
```

---

struct GtkWidgetClass

```
struct GtkWidgetClass {
  GInitiallyUnownedClass parent_class;


  guint activate_signal;

  /* seldomly overidden */
  void (*dispatch_child_properties_changed) (GtkWidget   *widget,
                                             guint        n_pspecs,
                                             GParamSpec **pspecs);

  /* basics */
  void (* destroy)             (GtkWidget        *widget);
  void (* show)                (GtkWidget        *widget);
  void (* show_all)            (GtkWidget        *widget);
  void (* hide)                (GtkWidget        *widget);
  void (* map)                 (GtkWidget        *widget);
  void (* unmap)         (GtkWidget        *widget);
  void (* realize)             (GtkWidget        *widget);
  void (* unrealize)           (GtkWidget        *widget);
  void (* size_allocate)       (GtkWidget        *widget,
                        GtkAllocation    *allocation);
  void (* state_changed)       (GtkWidget        *widget,
                        GtkStateType        previous_state);
  void (* state_flags_changed) (GtkWidget        *widget,
                        GtkStateFlags       previous_state_flags);
  void (* parent_set)          (GtkWidget        *widget,
                        GtkWidget        *previous_parent);
  void (* hierarchy_changed)   (GtkWidget        *widget,
                        GtkWidget        *previous_toplevel);
  void (* style_set)           (GtkWidget        *widget,
                        GtkStyle         *previous_style);
  void (* direction_changed)   (GtkWidget        *widget,
                        GtkTextDirection  previous_direction);
  void (* grab_notify)         (GtkWidget        *widget,
                        gboolean          was_grabbed);
  void (* child_notify)        (GtkWidget  *widget,
                        GParamSpec       *child_property);
  gboolean (* draw)            (GtkWidget  *widget,
                                cairo_t          *cr);

  /* size requests */
  GtkSizeRequestMode (* get_request_mode)              (GtkWidget      *widget);

  void               (* get_preferred_height)          (GtkWidget       *widget,
                                                        gint            *minimum_height,
                                                        gint            *natural_height);
  void               (* get_preferred_width_for_height) (GtkWidget       *widget,
                                                         gint             height,
                                                         gint            *minimum_width,
```

```
                                                    gint            *natural_width);
void              (* get_preferred_width)          (GtkWidget       *widget,
                                                    gint            *minimum_width,
                                                    gint            *natural_width);
void              (* get_preferred_height_for_width) (GtkWidget       *widget,
                                                    gint             width,
                                                    gint            *minimum_height,
                                                    gint            *natural_height);

/* Mnemonics */
gboolean (* mnemonic_activate)       (GtkWidget           *widget,
                                      gboolean             group_cycling);

/* explicit focus */
void     (* grab_focus)              (GtkWidget           *widget);
gboolean (* focus)                   (GtkWidget           *widget,
                                      GtkDirectionType     direction);

/* keyboard navigation */
void     (* move_focus)              (GtkWidget           *widget,
                                      GtkDirectionType     direction);
gboolean (* keynav_failed)           (GtkWidget           *widget,
                                      GtkDirectionType     direction);

/* events */
gboolean (* event)              (GtkWidget       *widget,
                                 GdkEvent        *event);
gboolean (* button_press_event) (GtkWidget       *widget,
                                 GdkEventButton      *event);
gboolean (* button_release_event) (GtkWidget      *widget,
                                 GdkEventButton      *event);
gboolean (* scroll_event)       (GtkWidget           *widget,
                                 GdkEventScroll      *event);
gboolean (* motion_notify_event) (GtkWidget       *widget,
                                 GdkEventMotion      *event);
gboolean (* delete_event)       (GtkWidget       *widget,
                                 GdkEventAny         *event);
gboolean (* destroy_event)      (GtkWidget       *widget,
                                 GdkEventAny         *event);
gboolean (* key_press_event)          (GtkWidget      *widget,
                                 GdkEventKey         *event);
gboolean (* key_release_event) (GtkWidget        *widget,
                                 GdkEventKey         *event);
gboolean (* enter_notify_event) (GtkWidget       *widget,
                                 GdkEventCrossing    *event);
gboolean (* leave_notify_event) (GtkWidget       *widget,
                                 GdkEventCrossing    *event);
gboolean (* configure_event)          (GtkWidget      *widget,
                                 GdkEventConfigure   *event);
gboolean (* focus_in_event)     (GtkWidget       *widget,
                                 GdkEventFocus       *event);
gboolean (* focus_out_event)          (GtkWidget      *widget,
                                 GdkEventFocus       *event);
gboolean (* map_event)          (GtkWidget       *widget,
                                 GdkEventAny         *event);
gboolean (* unmap_event)        (GtkWidget       *widget,
                                 GdkEventAny         *event);
gboolean (* property_notify_event) (GtkWidget     *widget,
                                 GdkEventProperty    *event);
gboolean (* selection_clear_event) (GtkWidget     *widget,
                                 GdkEventSelection   *event);
gboolean (* selection_request_event) (GtkWidget       *widget,
                                 GdkEventSelection   *event);
```

```
gboolean (* selection_notify_event) (GtkWidget          *widget,
                                     GdkEventSelection   *event);
gboolean (* proximity_in_event) (GtkWidget        *widget,
                                     GdkEventProximity   *event);
gboolean (* proximity_out_event) (GtkWidget       *widget,
                                     GdkEventProximity   *event);
gboolean (* visibility_notify_event) (GtkWidget         *widget,
                                     GdkEventVisibility  *event);
gboolean (* window_state_event) (GtkWidget        *widget,
                                     GdkEventWindowState *event);
gboolean (* damage_event)             (GtkWidget           *widget,
                                       GdkEventExpose      *event);
gboolean (* grab_broken_event)        (GtkWidget           *widget,
                                       GdkEventGrabBroken  *event);

/* selection */
void     (* selection_get)       (GtkWidget         *widget,
                                  GtkSelectionData   *selection_data,
                                  guint               info,
                                  guint               time_);
void     (* selection_received)  (GtkWidget         *widget,
                                  GtkSelectionData   *selection_data,
                                  guint               time_);

/* Source side drag signals */
void     (* drag_begin)          (GtkWidget        *widget,
                                  GdkDragContext     *context);
void     (* drag_end)            (GtkWidget         *widget,
                                  GdkDragContext     *context);
void     (* drag_data_get)       (GtkWidget         *widget,
                                  GdkDragContext     *context,
                                  GtkSelectionData   *selection_data,
                                  guint               info,
                                  guint               time_);
void     (* drag_data_delete)    (GtkWidget         *widget,
                                  GdkDragContext     *context);

/* Target side drag signals */
void     (* drag_leave)          (GtkWidget         *widget,
                                  GdkDragContext     *context,
                                  guint               time_);
gboolean (* drag_motion)         (GtkWidget        *widget,
                                  GdkDragContext     *context,
                                  gint                x,
                                  gint                y,
                                  guint               time_);
gboolean (* drag_drop)           (GtkWidget        *widget,
                                  GdkDragContext     *context,
                                  gint                x,
                                  gint                y,
                                  guint               time_);
void     (* drag_data_received)  (GtkWidget         *widget,
                                  GdkDragContext     *context,
                                  gint                x,
                                  gint                y,
                                  GtkSelectionData   *selection_data,
                                  guint               info,
                                  guint               time_);
gboolean (* drag_failed)         (GtkWidget          *widget,
                                  GdkDragContext     *context,
                                  GtkDragResult       result);

/* Signals used only for keybindings */
```

```
  gboolean (* popup_menu)          (GtkWidget          *widget);

  /* If a widget has multiple tooltips/whatsthis, it should show the
   * one for the current focus location, or if that doesn't make
   * sense, should cycle through them showing each tip alongside
   * whatever piece of the widget it applies to.
   */
  gboolean (* show_help)           (GtkWidget          *widget,
                                    GtkWidgetHelpType   help_type);

  /* accessibility support
   */
  AtkObject *  (* get_accessible)     (GtkWidget *widget);

  void         (* screen_changed)     (GtkWidget *widget,
                                       GdkScreen *previous_screen);
  gboolean     (* can_activate_accel) (GtkWidget *widget,
                                       guint      signal_id);


  void         (* composited_changed) (GtkWidget *widget);

  gboolean     (* query_tooltip)      (GtkWidget  *widget,
                                gint        x,
                                gint        y,
                                gboolean    keyboard_tooltip,
                                GtkTooltip *tooltip);

  void         (* compute_expand)     (GtkWidget  *widget,
                                       gboolean   *hexpand_p,
                                       gboolean   *vexpand_p);

  void         (* adjust_size_request)    (GtkWidget         *widget,
                                           GtkOrientation     orientation,
                                           gint              *minimum_size,
                                           gint              *natural_size);
  void         (* adjust_size_allocation) (GtkWidget         *widget,
                                           GtkOrientation     orientation,
                                           gint              *minimum_size,
                                           gint              *natural_size,
                                           gint              *allocated_pos,
                                           gint              *allocated_size);

  void         (* style_updated)          (GtkWidget *widget);

  gboolean     (* touch_event)            (GtkWidget     *widget,
                                           GdkEventTouch *event);

  void         (* get_preferred_height_and_baseline_for_width)  (GtkWidget     *widget,
                                                 gint           width,
                                                 gint          *minimum_height,
                                                 gint          *natural_height,
                                                 gint          *minimum_baseline,
                                                 gint          *natural_baseline);
  void         (* adjust_baseline_request)(GtkWidget         *widget,
                                           gint              *minimum_baseline,
                                           gint              *natural_baseline);
  void         (* adjust_baseline_allocation) (GtkWidget         *widget,
                                       gint              *baseline);
  void         (*queue_draw_region)           (GtkWidget         *widget,
                                       const cairo_region_t *region);
};
```

Members

guint `activate_signal`;	The signal to emit when a widget of this class is activated, `gtk_widget_activate()` handles the emission. Implementation of this signal is optional.
`dispatch_child_properties_changed` ()	Seldomly overidden.
`destroy` ()	Signals that all holders of a reference to the widget should release the reference that they hold.
`show` ()	Signal emitted when widget is shown
`show_all` ()	Recursively shows a widget, and any child widgets (if the widget is a container).
`hide` ()	Signal emitted when widget is hidden.
`map` ()	Signal emitted when widget is going to be mapped, that is when the widget is visible (which is controlled with `gtk_widget_set_visible()`) and all its parents up to the toplevel widget are also visible.
`unmap` ()	Signal emitted when widget is going to be unmapped, which means that either it or any of its parents up to the toplevel widget have been set as hidden.
`realize` ()	Signal emitted when widget is associated with a GdkWindow, which means that `gtk_widget_realize()` has been called or the widget has been mapped (that is, it is going to be drawn).
`unrealize` ()	Signal emitted when the GdkWindow associated with widget is destroyed, which means that `gtk_widget_unrealize()` has been called or the widget has been unmapped (that is, it is going to be hidden).
`size_allocate` ()	Signal emitted to get the widget allocation.
`state_changed` ()	Signal emitted when the widget state changes. Deprecated: 3.0
`state_flags_changed` ()	Signal emitted when the widget state changes, see `gtk_widget_get_state_flags()`.

`parent_set` ()	Signal emitted when a new parent has been set on a widget.
`hierarchy_changed` ()	Signal emitted when the anchored state of a widget changes.
`style_set` ()	Signal emitted when a new style has been set on a widget. Deprecated: 3.0
`direction_changed` ()	Signal emitted when the text direction of a widget changes.
`grab_notify` ()	Signal emitted when a widget becomes shadowed by a GTK+ grab (not a pointer or keyboard grab) on another widget, or when it becomes unshadowed due to a grab being removed.
`child_notify` ()	Signal emitted for each child property that has changed on an object.
`draw` ()	Signal emitted when a widget is supposed to render itself.
`get_request_mode` ()	This allows a widget to tell its parent container whether it prefers to be allocated in `GTK_SIZE_REQUEST_HEIGHT_FOR_WIDTH` or `GTK_SIZE_REQUEST_WIDTH_FOR_HEIGHT` mode. `GTK_SIZE_REQUEST_HEIGHT_FOR_WIDTH` means the widget prefers to have `GtkWidgetClass.get_preferred_width()` called and then `GtkWidgetClass.get_preferred_height_for_width()`. `GTK_SIZE_REQUEST_CONSTANT_SIZE` disables any height-for-width or width-for-height geometry management for a said widget and is the default return. It's important to note (as described below) that any widget which trades height-for-width or width-for-height must respond properly to both of the virtual methods `GtkWidgetClass.get_preferred_height_for_width()` and `GtkWidgetClass.get_preferred_width_for_height()` since it might be queried in either GtkSizeRequestMode by its parent container.
`get_preferred_height` ()	This is called by containers to obtain the minimum and natural

	height of a widget. A widget that does not actually trade any height for width or width for height only has to implement these two virtual methods (GtkWidgetClass.get_preferred_width() and GtkWidgetClass.get_preferred_height()).
get_preferred_width_for_height ()	This is analogous to GtkWidgetClass.get_preferred_height_for_width() except that it operates in the oposite orientation. It's rare that a widget actually does GTK_SIZE_REQUEST_WIDTH_FOR_HEIGHT requests but this can happen when, for example, a widget or container gets additional columns to compensate for a smaller allocated height.
get_preferred_width ()	This is called by containers to obtain the minimum and natural width of a widget. A widget will never be allocated a width less than its minimum and will only ever be allocated a width greater than the natural width once all of the said widget's siblings have received their natural widths. Furthermore, a widget will only ever be allocated a width greater than its natural width if it was configured to receive extra expand space from its parent container.
get_preferred_height_for_width ()	This is similar to GtkWidgetClass.get_preferred_height() except that it is passed a contextual width to request height for. By implementing this virtual method it is possible for a GtkLabel to tell its parent how much height would be required if the label were to be allocated a said width.
mnemonic_activate ()	Activates the widget if group_cycling is FALSE, and just grabs the focus if group_cycling is TRUE.
grab_focus ()	Causes widget to have the keyboard focus for the GtkWindow it's inside.
focus ()	
move_focus ()	Signal emitted when a change of

	focus is requested
`keynav_failed` ()	Signal emitted if keyboard navigation fails.
`event` ()	The GTK+ main loop will emit three signals for each GDK event delivered to a widget: one generic ::event signal, another, more specific, signal that matches the type of event delivered (e.g. "key-press-event") and finally a generic "event-after" signal.
`button_press_event` ()	Signal will be emitted when a button (typically from a mouse) is pressed.
`button_release_event` ()	Signal will be emitted when a button (typically from a mouse) is released.
`scroll_event` ()	Signal emitted when a button in the 4 to 7 range is pressed.
`motion_notify_event` ()	Signal emitted when the pointer moves over the widget's GdkWindow.
`delete_event` ()	Signal emitted if a user requests that a toplevel window is closed.
`destroy_event` ()	Signal is emitted when a GdkWindow is destroyed.
`key_press_event` ()	Signal emitted when a key is pressed.
`key_release_event` ()	Signal is emitted when a key is released.
`enter_notify_event` ()	Signal event will be emitted when the pointer enters the widget's window.
`leave_notify_event` ()	Will be emitted when the pointer leaves the widget's window.
`configure_event` ()	Signal will be emitted when the size, position or stacking of the widget's window has changed.
`focus_in_event` ()	Signal emitted when the keyboard focus enters the widget's window.
`focus_out_event` ()	Signal emitted when the keyboard focus leaves the widget's window.
`map_event` ()	Signal emitted when the widget's window is mapped.
`unmap_event` ()	Signal will be emitted when the widget's window is unmapped.
`property_notify_event` ()	Signal will be emitted when a property on the widget's window has been changed or deleted.
`selection_clear_event` ()	Signal will be emitted when the the widget's window has lost ownership of a selection.

`selection_request_event` ()	Signal will be emitted when another client requests ownership of the selection owned by the widget's window.
`selection_notify_event` ()	
`proximity_in_event` ()	
`proximity_out_event` ()	
`visibility_notify_event` ()	Signal emitted when the widget’s window is obscured or unobscured.
`window_state_event` ()	Signal emitted when the state of the toplevel window associated to the widget changes.
`damage_event` ()	Signal emitted when a redirected window belonging to widget gets drawn into.
`grab_broken_event` ()	Signal emitted when a pointer or keyboard grab on a window belonging to widget gets broken.
`selection_get` ()	
`selection_received` ()	
`drag_begin` ()	Signal emitted on the drag source when a drag is started.
`drag_end` ()	Signal emitted on the drag source when a drag is finished.
`drag_data_get` ()	Signal emitted on the drag source when the drop site requests the data which is dragged.
`drag_data_delete` ()	Signal emitted on the drag source when a drag with the action `GDK_ACTION_MOVE` is successfully completed.
`drag_leave` ()	Signal emitted on the drop site when the cursor leaves the widget.
`drag_motion` ()	signal emitted on the drop site when the user moves the cursor over the widget during a drag.
`drag_drop` ()	Signal emitted on the drop site when the user drops the data onto the widget.
`drag_data_received` ()	Signal emitted on the drop site when the dragged data has been received.
`drag_failed` ()	Signal emitted on the drag source when a drag has failed.
`popup_menu` ()	Signal emitted whenever a widget should pop up a context menu.
`show_help` ()	
`get_accessible` ()	Returns the accessible object that describes the widget to an assistive technology.
`screen_changed` ()	Signal emitted when the screen of a

	widget has changed.
`can_activate_accel` ()	Signal allows applications and derived widgets to override the default GtkWidget handling for determining whether an accelerator can be activated.
`composited_changed` ()	Signal emitted when the composited status of widgets screen changes. See `gdk_screen_is_composited()`.
`query_tooltip` ()	Signal emitted when “has-tooltip” is `TRUE` and the hover timeout has expired with the cursor hovering “above” widget; or emitted when widget got focus in keyboard mode.
`compute_expand` ()	Computes whether a container should give this widget extra space when possible.
`adjust_size_request` ()	Convert an initial size request from a widget's GtkSizeRequestMode virtual method implementations into a size request to be used by parent containers in laying out the widget. adjust_size_request adjusts from a child widget's original request to what a parent container should use for layout. The `for_size` argument will be -1 if the request should not be for a particular size in the opposing orientation, i.e. if the request is not height-for-width or width-for-height. If `for_size` is greater than -1, it is the proposed allocation in the opposing orientation that we need the request for. Implementations of adjust_size_request should chain up to the default implementation, which applies GtkWidget’s margin properties and imposes any values from `gtk_widget_set_size_request()`. Chaining up should be last, after your subclass adjusts the request, so GtkWidget can apply constraints and add the margin properly.
`adjust_size_allocation` ()	Convert an initial size allocation assigned by a GtkContainer using `gtk_widget_size_allocate()`, into an actual size allocation to be used by the widget. adjust_size_allocation adjusts to a

	child widget's actual allocation from what a parent container computed for the child. The adjusted allocation must be entirely within the original allocation. In any custom implementation, chain up to the default GtkWidget implementation of this method, which applies the margin and alignment properties of GtkWidget. Chain up before performing your own adjustments so your own adjustments remove more allocation after the GtkWidget base class has already removed margin and alignment. The natural size passed in should be adjusted in the same way as the allocated size, which allows adjustments to perform alignments or other changes based on natural size.
`style_updated` ()	Signal emitted when the GtkStyleContext of a widget is changed.
`touch_event` ()	Signal emitted when a touch event happens
`get_preferred_height_and_baseline_for_width` ()	
`adjust_baseline_request` ()	
`adjust_baseline_allocation` ()	
`queue_draw_region` ()	Invalidates the area of widget defined by region by calling `gdk_window_invalidate_region()` on the widget's window and all its child windows.

GtkRequisition

```
typedef struct {
  gint width;
  gint height;
} GtkRequisition;
```

A GtkRequisition represents the desired size of a widget. See GtkWidget's geometry management section for more information.

Members

gint `width`;	the widget's desired width
gint `height`;	the widget's desired height

GtkAllocation

```
typedef     GdkRectangle        GtkAllocation;
```

A GtkAllocation of a widget represents region which has been allocated to the widget by its parent. It is a subregion of its parents allocation. See GtkWidget's geometry management section for more information.

---

enum GtkWidgetHelpType

Kinds of widget-specific help. Used by the ::show-help signal.

Members

GTK_WIDGET_HELP_TOOLTIP	Tooltip.
GTK_WIDGET_HELP_WHATS_THIS	What's this.

---

enum GtkTextDirection

Reading directions for text.

Members

GTK_TEXT_DIR_NONE	No direction.
GTK_TEXT_DIR_LTR	Left to right text direction.
GTK_TEXT_DIR_RTL	Right to left text direction.

---

enum GtkStateType

`GtkStateType` has been deprecated since version 3.14 and should not be used in newly-written code.

All APIs that are using this enumeration have been deprecated in favor of alternatives using GtkStateFlags.

This type indicates the current state of a widget; the state determines how the widget is drawn. The GtkStateType enumeration is also used to identify different colors in a GtkStyle for drawing, so states can be used for subparts of a widget as well as entire widgets.

Members

GTK_STATE_NORMAL	State during normal operation.
GTK_STATE_ACTIVE	State of a currently active widget, such as a depressed button.
GTK_STATE_PRELIGHT	State indicating that the mouse

	pointer is over the widget and the widget will respond to mouse clicks.
GTK_STATE_SELECTED	State of a selected item, such the selected row in a list.
GTK_STATE_INSENSITIVE	State indicating that the widget is unresponsive to user actions.
GTK_STATE_INCONSISTENT	The widget is inconsistent, such as checkbuttons or radiobuttons that aren't either set to `TRUE` nor `FALSE`, or buttons requiring the user attention.
GTK_STATE_FOCUSED	The widget has the keyboard focus.

enum GtkSizeRequestMode

Specifies a preference for height-for-width or width-for-height geometry management.

Members

GTK_SIZE_REQUEST_HEIGHT_FOR_WIDTH	Prefer height-for-width geometry management
GTK_SIZE_REQUEST_WIDTH_FOR_HEIGHT	Prefer width-for-height geometry management
GTK_SIZE_REQUEST_CONSTANT_SIZE	Don't trade height-for-width or width-for-height

struct GtkRequestedSize

```
struct GtkRequestedSize {
  gpointer data;
  gint     minimum_size;
  gint     natural_size;
};
```

Represents a request of a screen object in a given orientation. These are primarily used in container implementations when allocating a natural size for children calling. See `gtk_distribute_natural_allocation()`.

Members

gpointer `data`;	A client pointer
gint `minimum_size`;	The minimum size needed for allocation in a given orientation
gint `natural_size`;	The natural size for allocation in a given orientation

enum GtkAlign

Controls how a widget deals with extra space in a single (x or y) dimension.

Alignment only matters if the widget receives a “too large” allocation, for example if you packed the widget with the “expand” flag inside a GtkBox, then the widget might get extra space. If you have for example a 16x16 icon inside a 32x32 space, the icon could be scaled and stretched, it could be centered, or it could be positioned to one side of the space.

Note that in horizontal context `GTK_ALIGN_START` and `GTK_ALIGN_END` are interpreted relative to text direction.

GTK_ALIGN_BASELINE support for it is optional for containers and widgets, and it is only supported for vertical alignment. When its not supported by a child or a container it is treated as `GTK_ALIGN_FILL` .

Members

GTK_ALIGN_FILL	stretch to fill all space if possible, center if no meaningful way to stretch
GTK_ALIGN_START	snap to left or top side, leaving space on right or bottom
GTK_ALIGN_END	snap to right or bottom side, leaving space on left or top
GTK_ALIGN_CENTER	center natural width of widget inside the allocation
GTK_ALIGN_BASELINE	align the widget according to the baseline. Since 3.10.

Property Details

The `“app-paintable”` property

`“app-paintable”` `gboolean`

Whether the application will paint directly on the widget.

Flags: Read / Write

Default value: FALSE

The `“can-default”` property

`“can-default”` `gboolean`

Whether the widget can be the default widget.

Flags: Read / Write

Default value: FALSE

The “can-focus” property

“can-focus” gboolean

Whether the widget can accept the input focus.

Flags: Read / Write

Default value: FALSE

The “composite-child” property

“composite-child” gboolean

Whether the widget is part of a composite widget.

Flags: Read

Default value: FALSE

The “double-buffered” property

“double-buffered” gboolean

Whether the widget is double buffered.

GtkWidget:double-buffered has been deprecated since version 3.14 and should not be used in newly-written code.

Widgets should not use this property.

Flags: Read / Write

Default value: TRUE

Since: 2.18

The “events” property

“events” GdkEventMask

The event mask that decides what kind of GdkEvents this widget gets.

Flags: Read / Write

Default value: GDK_STRUCTURE_MASK

The “expand” property

“expand” gboolean

Whether to expand in both directions. Setting this sets both “hexpand” and “vexpand”

Flags: Read / Write

Default value: FALSE

Since: 3.0

The “focus-on-click” property

“focus-on-click” gboolean

Whether the widget should grab focus when it is clicked with the mouse.

This property is only relevant for widgets that can take focus.

Before 3.20, several widgets (GtkButton, GtkFileChooserButton, GtkComboBox) implemented this property individually.

Flags: Read / Write

Default value: TRUE

Since: 3.20

The “halign” property

“halign” GtkAlign

How to distribute horizontal space if widget gets extra space, see GtkAlign

Flags: Read / Write

Default value: GTK_ALIGN_FILL

Since: 3.0

The “has-default” property

“has-default” gboolean

Whether the widget is the default widget.

Flags: Read / Write

Default value: FALSE

The “has-focus” property

“has-focus” gboolean

Whether the widget has the input focus.

Flags: Read / Write

Default value: FALSE

The “has-tooltip” property

“has-tooltip” gboolean

Enables or disables the emission of “query-tooltip” on widget . A value of TRUE indicates that widget can have a tooltip, in this case the widget will be queried using “query-tooltip” to determine whether it will provide a tooltip or not.

Note that setting this property to TRUE for the first time will change the event masks of the GdkWindows of this widget to include leave-notify and motion-notify events. This cannot and will not be undone when the property is set to FALSE again.

Flags: Read / Write

Default value: FALSE

Since: 2.12

The “height-request” property

“height-request” gint

Override for height request of the widget, or -1 if natural request should be used.

Flags: Read / Write

Allowed values: >= -1

Default value: -1

The “hexpand” property

“hexpand” gboolean

Whether to expand horizontally. See gtk_widget_set_hexpand().

Flags: Read / Write

Default value: FALSE

Since: 3.0

The “hexpand-set” property

“hexpand-set” gboolean

Whether to use the “hexpand” property. See gtk_widget_get_hexpand_set().

Flags: Read / Write

Default value: FALSE

Since: 3.0

The “is-focus” property

“is-focus” gboolean

Whether the widget is the focus widget within the toplevel.

Flags: Read / Write

Default value: FALSE

The “margin” property

“margin” gint

Sets all four sides' margin at once. If read, returns max margin on any side.

Flags: Read / Write

Allowed values: [0,32767]

Default value: 0

Since: 3.0

The “margin-bottom” property

“margin-bottom” gint

Margin on bottom side of widget.

This property adds margin outside of the widget's normal size request, the margin will be added in addition to the size from `gtk_widget_set_size_request()` for example.

Flags: Read / Write

Allowed values: [0,32767]

Default value: 0

Since: 3.0

The “margin-end” property

“margin-end” gint

Margin on end of widget, horizontally. This property supports left-to-right and right-to-left text directions.

This property adds margin outside of the widget's normal size request, the margin will be added in addition to the size from `gtk_widget_set_size_request()` for example.

Flags: Read / Write

Allowed values: [0,32767]

Default value: 0

Since: 3.12

The “margin-left” property

“margin-left” gint

Margin on left side of widget.

This property adds margin outside of the widget's normal size request, the margin will be added in addition to the size from `gtk_widget_set_size_request()` for example.

`GtkWidget:margin-left` has been deprecated since version 3.12 and should not be used in newly-written code.

Use “margin-start” instead.

Flags: Read / Write

Allowed values: [0,32767]

Default value: 0

Since: 3.0

The “margin-right” property

“margin-right” gint

Margin on right side of widget.

This property adds margin outside of the widget's normal size request, the margin will be added in addition to the size from `gtk_widget_set_size_request()` for example.

`GtkWidget:margin-right` has been deprecated since version 3.12 and should not be used in newly-written code.

Use “margin-end” instead.

Flags: Read / Write

Allowed values: [0,32767]

Default value: 0

Since: 3.0

The “margin-start” property

“margin-start” gint

Margin on start of widget, horizontally. This property supports left-to-right and right-to-left text directions.

This property adds margin outside of the widget's normal size request, the margin will be added in addition to the size from `gtk_widget_set_size_request()` for example.

Flags: Read / Write

Allowed values: [0,32767]

Default value: 0

Since: 3.12

The “margin-top” property

`“margin-top”` `gint`

Margin on top side of widget.

This property adds margin outside of the widget's normal size request, the margin will be added in addition to the size from `gtk_widget_set_size_request()` for example.

Flags: Read / Write

Allowed values: [0,32767]

Default value: 0

Since: 3.0

The “name” property

`“name”` `gchar *`

The name of the widget.

Flags: Read / Write

Default value: NULL

The “no-show-all” property

`“no-show-all”` `gboolean`

Whether gtk_widget_show_all() should not affect this widget.

Flags: Read / Write

Default value: FALSE

The “opacity” property

“opacity” gdouble

The requested opacity of the widget. See gtk_widget_set_opacity() for more details about window opacity.

Before 3.8 this was only available in GtkWindow

Flags: Read / Write

Allowed values: [0,1]

Default value: 1

Since: 3.8

The “parent” property

“parent” GtkContainer *

The parent widget of this widget. Must be a Container widget.

Flags: Read / Write

The “receives-default” property

“receives-default” gboolean

If TRUE, the widget will receive the default action when it is focused.

Flags: Read / Write

Default value: FALSE

The “scale-factor” property

“scale-factor” gint

The scale factor of the widget. See gtk_widget_get_scale_factor() for more details about widget scaling.

Flags: Read

Allowed values: >= 1

Default value: 1

Since: 3.10

The “sensitive” property

“sensitive” gboolean

Whether the widget responds to input.

Flags: Read / Write

Default value: TRUE

The “style” property

“style” GtkStyle *

The style of the widget, which contains information about how it will look (colors, etc).

GtkWidget:style is deprecated and should not be used in newly-written code.

Use GtkStyleContext instead

Flags: Read / Write

The “tooltip-markup” property

“tooltip-markup” gchar *

Sets the text of tooltip to be the given string, which is marked up with the Pango text markup language. Also see gtk_tooltip_set_markup().

This is a convenience property which will take care of getting the tooltip shown if the given string is not NULL: “has-tooltip” will automatically be set to TRUE and there will be taken care of “query-tooltip” in the default signal handler.

Note that if both “tooltip-text” and “tooltip-markup” are set, the last one wins.

Flags: Read / Write

Default value: NULL

Since: 2.12

The “tooltip-text” property

“tooltip-text” gchar *

Sets the text of tooltip to be the given string.

Also see gtk_tooltip_set_text().

This is a convenience property which will take care of getting the tooltip shown if the given string is not NULL: “has-tooltip” will automatically be set to TRUE and there will be taken care of “query-tooltip” in the default signal handler.

Note that if both “tooltip-text” and “tooltip-markup” are set, the last one wins.

Flags: Read / Write

Default value: NULL

Since: 2.12

The “valign” property

“valign” GtkAlign

How to distribute vertical space if widget gets extra space, see GtkAlign

Flags: Read / Write

Default value: GTK_ALIGN_FILL

Since: 3.0

The “vexpand” property

“vexpand” gboolean

Whether to expand vertically. See gtk_widget_set_vexpand().

Flags: Read / Write

Default value: FALSE

Since: 3.0

The “vexpand-set” property

“vexpand-set” gboolean

Whether to use the “vexpand” property. See gtk_widget_get_vexpand_set().

Flags: Read / Write

Default value: FALSE

Since: 3.0

The “visible” property

“visible” gboolean

Whether the widget is visible.

Flags: Read / Write

Default value: FALSE

The “width-request” property

“width-request” gint

Override for width request of the widget, or -1 if natural request should be used.

Flags: Read / Write

Allowed values: >= -1

Default value: -1

The “window” property

“window” GdkWindow *

The widget's window if it is realized, NULL otherwise.

Flags: Read

Since: 2.14

Style Property Details

The “cursor-aspect-ratio” style property

“cursor-aspect-ratio” gfloat

Aspect ratio with which to draw insertion cursor.

Flags: Read

Allowed values: [0,1]

Default value: 0.04

The “cursor-color” style property

“cursor-color” GdkColor *

The color with which to draw the insertion cursor in entries and text views.

GtkWidget:cursor-color has been deprecated since version 3.20 and should not be used in newly-written code.

Use the caret-color CSS property

Flags: Read

The “focus-line-pattern” style property

“focus-line-pattern” gchar *

The "focus-line-pattern" style property defines the dash pattern used to draw the focus indicator. The character values are interpreted as pixel widths of alternating on and off segments of the line.

GtkWidget:focus-line-pattern has been deprecated since version 3.14 and should not be used in newly-written code.

use the outline-style CSS property instead.

Flags: Read

Default value: "\001\001"

The “focus-line-width” style property

“focus-line-width” gint

The "focus-line-width" style property defines the width, in pixels, of the focus indicator line

GtkWidget:focus-line-width has been deprecated since version 3.14 and should not be used in newly-written code.

use the outline-width and padding CSS properties instead.

Flags: Read

Allowed values: >= 0

Default value: 1

The “focus-padding” style property

“focus-padding” gint

The "focus-padding" style property defines the width, in pixels, between focus indicator and the widget 'box'.

GtkWidget:focus-padding has been deprecated since version 3.14 and should not be used in newly-written code.

use the outline-offset CSS properties instead.

Flags: Read

Allowed values: >= 0

Default value: 1

The “interior-focus” style property

“interior-focus” gboolean

The "interior-focus" style property defines whether to draw the focus indicator inside widgets.

GtkWidget:interior-focus has been deprecated since version 3.14 and should not be used in newly-written code.

use the outline CSS properties instead.

Flags: Read

Default value: TRUE

The “link-color” style property

“link-color” GdkColor *

The "link-color" style property defines the color of unvisited links.

GtkWidget:link-color has been deprecated since version 3.12 and should not be used in newly-written code.

Links now use a separate state flags for selecting different theming, this style property is ignored

Flags: Read

Since: 2.10

The “scroll-arrow-hlength” style property

“scroll-arrow-hlength” gint

The "scroll-arrow-hlength" style property defines the length of horizontal scroll arrows.

Flags: Read

Allowed values: >= 1

Default value: 16

Since: 2.10

The “scroll-arrow-vlength” style property

“scroll-arrow-vlength” gint

The "scroll-arrow-vlength" style property defines the length of vertical scroll arrows.

Flags: Read

Allowed values: >= 1

Default value: 16

Since: 2.10

The “secondary-cursor-color” style property

“secondary-cursor-color” GdkColor *

The color with which to draw the secondary insertion cursor in entries and text views when editing mixed right-to-left and left-to-right text.

GtkWidget:secondary-cursor-color has been deprecated since version 3.20 and should not be used in newly-written code.

Use the -gtk-secondary-caret-color CSS property

Flags: Read

The “separator-height” style property

“separator-height” gint

The "separator-height" style property defines the height of separators. This property only takes effect if the "wide-separators" style property is TRUE.

GtkWidget:separator-height has been deprecated since version 3.20 and should not be used in newly-written code.

Use the standard min-height CSS property on the separator elements to size separators; the value of this style property is ignored.

Flags: Read

Allowed values: >= 0

Default value: 0

Since: 2.10

The “separator-width” style property

“separator-width” gint

The "separator-width" style property defines the width of separators. This property only takes effect if the "wide-separators" style property is TRUE.

GtkWidget:separator-width has been deprecated since version 3.20 and should not be used in newly-written code.

Use the standard min-width CSS property on the separator elements to size separators; the value of this style property is ignored.

Flags: Read

Allowed values: >= 0

Default value: 0

Since: 2.10

The “text-handle-height” style property

“text-handle-height” gint

Height of text selection handles.

Flags: Read

Allowed values: >= 1

Default value: 20

The “text-handle-width” style property

“text-handle-width” gint

Width of text selection handles.

Flags: Read

Allowed values: >= 1

Default value: 16

The “visited-link-color” style property

“visited-link-color” GdkColor *

The "visited-link-color" style property defines the color of visited links.

GtkWidget:visited-link-color has been deprecated since version 3.12 and should not be used in newly-written code.

Links now use a separate state flags for selecting different theming, this style property is ignored

Flags: Read

Since: 2.10

The “wide-separators” style property

“wide-separators” gboolean

The "wide-separators" style property defines whether separators have configurable width and should be drawn using a box instead of a line.

GtkWidget:wide-separators has been deprecated since version 3.20 and should not be used in newly-written code.

Use CSS properties on the separator elements to style separators; the value of this style property is ignored.

Flags: Read

Default value: FALSE

Since: 2.10

The “window-dragging” style property

“window-dragging” gboolean

Whether windows can be dragged and maximized by clicking on empty areas.

Flags: Read

Default value: FALSE

Signal Details

The `"accel-closures-changed"` signal

```
void
user_function (GtkWidget *widget,
               gpointer   user_data)
```

Parameters

widget	the object which received the signal.	
user_data	user data set when the signal handler was connected.	

The `"button-press-event"` signal

```
gboolean
user_function (GtkWidget *widget,
               GdkEvent  *event,
               gpointer   user_data)
```

The ::button-press-event signal will be emitted when a button (typically from a mouse) is pressed.

To receive this signal, the GdkWindow associated to the widget needs to enable the GDK_BUTTON_PRESS_MASK mask.

This signal will be sent to the grab widget if there is one.

Parameters

widget	the object which received the signal.	
event	the GdkEventButton which triggered this signal.	[type Gdk.EventButton]
user_data	user data set when the signal handler was connected.	

Returns

`TRUE` to stop other handlers from being invoked for the event. `FALSE` to propagate the event further.

Flags: Run Last

The `"button-release-event"` signal

```
gboolean
user_function (GtkWidget *widget,
               GdkEvent  *event,
               gpointer   user_data)
```

The ::button-release-event signal will be emitted when a button (typically from a mouse) is released.

To receive this signal, the GdkWindow associated to the widget needs to enable the GDK_BUTTON_RELEASE_MASK mask.

This signal will be sent to the grab widget if there is one.

Parameters

widget	the object which received the signal.	
event	the GdkEventButton which triggered this signal.	[type Gdk.EventButton]
user_data	user data set when the signal handler was connected.	

Returns

TRUE to stop other handlers from being invoked for the event. FALSE to propagate the event further.

Flags: Run Last

The “can-activate-accel” signal

```
gboolean
user_function (GtkWidget *widget,
               guint      signal_id,
               gpointer   user_data)
```

Determines whether an accelerator that activates the signal identified by signal_id can currently be activated. This signal is present to allow applications and derived widgets to override the default GtkWidget handling for determining whether an accelerator can be activated.

Parameters

widget	the object which received the signal
signal_id	the ID of a signal installed on widget
user_data	user data set when the signal handler was connected.

Returns

TRUE if the signal can be activated.

Flags: Run Last

The “child-notify” signal

```
void
user_function (GtkWidget  *widget,
```

```
               GParamSpec *child_property,
               gpointer    user_data)
```

The ::child-notify signal is emitted for each child property that has changed on an object. The signal's detail holds the property name.

Parameters

widget	the object which received the signal
child_property	the GParamSpec of the changed child property
user_data	user data set when the signal handler was connected.

Flags: No Hooks

The `"composited-changed"` signal

```
void
user_function (GtkWidget *widget,
               gpointer   user_data)
```

The ::composited-changed signal is emitted when the composited status of `widgets` screen changes. See `gdk_screen_is_composited()`.

`GtkWidget::composited-changed` has been deprecated since version 3.22 and should not be used in newly-written code.

Use GdkScreen::composited-changed instead.

Parameters

widget	the object on which the signal is emitted
user_data	user data set when the signal handler was connected.

Flags: Action

The `"configure-event"` signal

```
gboolean
user_function (GtkWidget *widget,
               GdkEvent  *event,
               gpointer   user_data)
```

The ::configure-event signal will be emitted when the size, position or stacking of the `widget` 's window has changed.

To receive this signal, the GdkWindow associated to the widget needs to enable the GDK_STRUCTURE_MASK mask. GDK will enable this mask automatically for all new windows.

Parameters

widget	the object which received the signal	
event	the GdkEventConfigure which triggered this signal.	[type Gdk.EventConfigure]
user_data	user data set when the signal handler was connected.	

Returns

`TRUE` to stop other handlers from being invoked for the event. `FALSE` to propagate the event further.

Flags: Run Last

The `"damage-event"` signal

```
gboolean
user_function (GtkWidget *widget,
               GdkEvent  *event,
               gpointer   user_data)
```

Emitted when a redirected window belonging to `widget` gets drawn into. The region/area members of the event shows what area of the redirected drawable was drawn into.

Parameters

widget	the object which received the signal	
event	the GdkEventExpose event.	[type Gdk.EventExpose]
user_data	user data set when the signal handler was connected.	

Returns

`TRUE` to stop other handlers from being invoked for the event. `FALSE` to propagate the event further.

Flags: Run Last

Since: 2.14

The `"delete-event"` signal

```
gboolean
user_function (GtkWidget *widget,
               GdkEvent  *event,
               gpointer   user_data)
```

The ::delete-event signal is emitted if a user requests that a toplevel window is closed. The default handler for this signal destroys the window. Connecting `gtk_widget_hide_on_delete()` to this signal will cause the window to be hidden instead, so that it can later be shown again without reconstructing it.

Parameters

widget	the object which received the signal
event	the event which triggered this signal
user_data	user data set when the signal handler was connected.

Returns

TRUE to stop other handlers from being invoked for the event. FALSE to propagate the event further.

Flags: Run Last

The “destroy” signal

```
void
user_function (GtkWidget *object,
               gpointer   user_data)
```

Signals that all holders of a reference to the widget should release the reference that they hold. May result in finalization of the widget if all references are released.

This signal is not suitable for saving widget state.

Parameters

object	the object which received the signal
user_data	user data set when the signal handler was connected.

Flags: No Hooks

The “destroy-event” signal

```
gboolean
user_function (GtkWidget *widget,
               GdkEvent  *event,
               gpointer   user_data)
```

The ::destroy-event signal is emitted when a GdkWindow is destroyed. You rarely get this signal, because most widgets disconnect themselves from their window before they destroy it, so no widget owns the window at destroy time.

To receive this signal, the GdkWindow associated to the widget needs to enable the GDK_STRUCTURE_MASK mask. GDK will enable this mask automatically for all new windows.

Parameters

widget	the object which received the signal.
event	the event which triggered this signal
user_data	user data set when the signal

handler was connected.

Returns

TRUE to stop other handlers from being invoked for the event. FALSE to propagate the event further.

Flags: Run Last

The “direction-changed” signal

```
void
user_function (GtkWidget       *widget,
               GtkTextDirection previous_direction,
               gpointer         user_data)
```

The ::direction-changed signal is emitted when the text direction of a widget changes.

Parameters

widget	the object on which the signal is emitted
previous_direction	the previous text direction of `widget`
user_data	user data set when the signal handler was connected.

Flags: Run First

The “drag-begin” signal

```
void
user_function (GtkWidget      *widget,
               GdkDragContext *context,
               gpointer        user_data)
```

The ::drag-begin signal is emitted on the drag source when a drag is started. A typical reason to connect to this signal is to set up a custom drag icon with e.g. `gtk_drag_source_set_icon_pixbuf()`.

Note that some widgets set up a drag icon in the default handler of this signal, so you may have to use `g_signal_connect_after()` to override what the default handler did.

Parameters

widget	the object which received the signal
context	the drag context
user_data	user data set when the signal handler was connected.

Flags: Run Last

The “drag-data-delete” signal

```
void
user_function (GtkWidget      *widget,
               GdkDragContext *context,
               gpointer        user_data)
```

The ::drag-data-delete signal is emitted on the drag source when a drag with the action GDK_ACTION_MOVE is successfully completed. The signal handler is responsible for deleting the data that has been dropped. What "delete" means depends on the context of the drag operation.

Parameters

widget	the object which received the signal
context	the drag context
user_data	user data set when the signal handler was connected.

Flags: Run Last

The “drag-data-get” signal

```
void
user_function (GtkWidget        *widget,
               GdkDragContext   *context,
               GtkSelectionData *data,
               guint             info,
               guint             time,
               gpointer          user_data)
```

The ::drag-data-get signal is emitted on the drag source when the drop site requests the data which is dragged. It is the responsibility of the signal handler to fill `data` with the data in the format which is indicated by `info` . See `gtk_selection_data_set()` and `gtk_selection_data_set_text()`.

Parameters

widget	the object which received the signal
context	the drag context
data	the GtkSelectionData to be filled with the dragged data
info	the info that has been registered with the target in the GtkTargetList
time	the timestamp at which the data was requested
user_data	user data set when the signal handler was connected.

Flags: Run Last

The “drag-data-received” signal

```
void
```

```
user_function (GtkWidget        *widget,
               GdkDragContext   *context,
               gint              x,
               gint              y,
               GtkSelectionData *data,
               guint             info,
               guint             time,
               gpointer          user_data)
```

The ::drag-data-received signal is emitted on the drop site when the dragged data has been received. If the data was received in order to determine whether the drop will be accepted, the handler is expected to call `gdk_drag_status()` and not finish the drag. If the data was received in response to a "drag-drop" signal (and this is the last target to be received), the handler for this signal is expected to process the received data and then call `gtk_drag_finish()`, setting the `success` parameter depending on whether the data was processed successfully.

Applications must create some means to determine why the signal was emitted and therefore whether to call `gdk_drag_status()` or `gtk_drag_finish()`.

The handler may inspect the selected action with `gdk_drag_context_get_selected_action()` before calling `gtk_drag_finish()`, e.g. to implement `GDK_ACTION_ASK` as shown in the following example:

```
1                                                 void
2                                                 drag_data_received (GtkWidget
3                                                 *widget,
4                                                                     GdkDragContext
5                                                 *context,
6                                                                     gint                x,
7                                                                     gint                y,
8                                                                     GtkSelectionData   *data,
9                                                                     guint               info,
10                                                                    guint               time)
11                                                {
12                                                  if ((data->length >= 0) && (data->format ==
13                                                8))
14                                                    {
15                                                      GdkDragAction action;
16
17                                                      // handle data here
18
19                                                      action =
20                                                gdk_drag_context_get_selected_action
21                                                (context);
22                                                      if (action == GDK_ACTION_ASK)
23                                                        {
24                                                          GtkWidget *dialog;
25                                                          gint response;
26
27                                                          dialog = gtk_message_dialog_new
28                                                (NULL,
29
30                                                GTK_DIALOG_MODAL |
31
32                                                GTK_DIALOG_DESTROY_WITH_PARENT,
33
34                                                GTK_MESSAGE_INFO,
35
36                                                GTK_BUTTONS_YES_NO,
37
38                                                "Move the data ?\n");
39                                                          response = gtk_dialog_run
40                                                (GTK_DIALOG (dialog));
41                                                          gtk_widget_destroy (dialog);
```

```
          if (response == GTK_RESPONSE_YES)
            action = GDK_ACTION_MOVE;
          else
            action = GDK_ACTION_COPY;
        }

      gtk_drag_finish (context, TRUE, action
== GDK_ACTION_MOVE, time);
    }
  else
    gtk_drag_finish (context, FALSE, FALSE,
time);
 }
```

Parameters

widget	the object which received the signal
context	the drag context
x	where the drop happened
y	where the drop happened
data	the received data
info	the info that has been registered with the target in the GtkTargetList
time	the timestamp at which the data was received
user_data	user data set when the signal handler was connected.

Flags: Run Last

The "drag-drop" signal

```
gboolean
user_function (GtkWidget      *widget,
               GdkDragContext *context,
               gint            x,
               gint            y,
               guint           time,
               gpointer        user_data)
```

The ::drag-drop signal is emitted on the drop site when the user drops the data onto the widget. The signal handler must determine whether the cursor position is in a drop zone or not. If it is not in a drop zone, it returns FALSE and no further processing is necessary. Otherwise, the handler returns TRUE. In this case, the handler must ensure that gtk_drag_finish() is called to let the source know that the drop is done. The call to gtk_drag_finish() can be done either directly or in a "drag-data-received" handler which gets triggered by calling gtk_drag_get_data() to receive the data for one or more of the supported targets.

Parameters

widget	the object which received the signal
context	the drag context
x	the x coordinate of the current cursor position

y	the y coordinate of the current cursor position
time	the timestamp of the motion event
user_data	user data set when the signal handler was connected.

Returns

whether the cursor position is in a drop zone

Flags: Run Last

The `"drag-end"` signal

```
void
user_function (GtkWidget      *widget,
               GdkDragContext *context,
               gpointer        user_data)
```

The ::drag-end signal is emitted on the drag source when a drag is finished. A typical reason to connect to this signal is to undo things done in "drag-begin".

Parameters

widget	the object which received the signal
context	the drag context
user_data	user data set when the signal handler was connected.

Flags: Run Last

The `"drag-failed"` signal

```
gboolean
user_function (GtkWidget      *widget,
               GdkDragContext *context,
               GtkDragResult   result,
               gpointer        user_data)
```

The ::drag-failed signal is emitted on the drag source when a drag has failed. The signal handler may hook custom code to handle a failed DnD operation based on the type of error, it returns `TRUE` is the failure has been already handled (not showing the default "drag operation failed" animation), otherwise it returns `FALSE`.

Parameters

widget	the object which received the signal
context	the drag context
result	the result of the drag operation
user_data	user data set when the signal handler was connected.

Returns

TRUE if the failed drag operation has been already handled.

Flags: Run Last

Since: 2.12

The “drag-leave” signal

```
void
user_function (GtkWidget      *widget,
               GdkDragContext *context,
               guint           time,
               gpointer        user_data)
```

The ::drag-leave signal is emitted on the drop site when the cursor leaves the widget. A typical reason to connect to this signal is to undo things done in “drag-motion”, e.g. undo highlighting with `gtk_drag_unhighlight()`.

Likewise, the “drag-leave” signal is also emitted before the ::drag-drop signal, for instance to allow cleaning up of a preview item created in the “drag-motion” signal handler.

Parameters

widget	the object which received the signal.
context	the drag context
time	the timestamp of the motion event
user_data	user data set when the signal handler was connected.

Flags: Run Last

The “drag-motion” signal

```
gboolean
user_function (GtkWidget      *widget,
               GdkDragContext *context,
               gint            x,
               gint            y,
               guint           time,
               gpointer        user_data)
```

The ::drag-motion signal is emitted on the drop site when the user moves the cursor over the widget during a drag. The signal handler must determine whether the cursor position is in a drop zone or not. If it is not in a drop zone, it returns FALSE and no further processing is necessary. Otherwise, the handler returns TRUE. In this case, the handler is responsible for providing the necessary information for displaying feedback to the user, by calling `gdk_drag_status()`.

If the decision whether the drop will be accepted or rejected can't be made based solely on the cursor position and the type of the data, the handler may inspect the dragged data by calling `gtk_drag_get_data()` and defer the `gdk_drag_status()` call to the “drag-data-received” handler. Note that you must pass GTK_DEST_DEFAULT_DROP, GTK_DEST_DEFAULT_MOTION or GTK_DEST_DEFAULT_ALL to

gtk_drag_dest_set() when using the drag-motion signal that way.

Also note that there is no drag-enter signal. The drag receiver has to keep track of whether he has received any drag-motion signals since the last "drag-leave" and if not, treat the drag-motion signal as an "enter" signal. Upon an "enter", the handler will typically highlight the drop site with gtk_drag_highlight().

```
static void
drag_motion (GtkWidget      *widget,
             GdkDragContext *context,
             gint            x,
             gint            y,
             guint           time)
{
  GdkAtom target;

  PrivateData *private_data =
GET_PRIVATE_DATA (widget);

  if (!private_data->drag_highlight)
   {
     private_data->drag_highlight = 1;
     gtk_drag_highlight (widget);
   }

  target = gtk_drag_dest_find_target (widget,
context, NULL);
  if (target == GDK_NONE)
    gdk_drag_status (context, 0, time);
  else
   {
     private_data->pending_status
        =
gdk_drag_context_get_suggested_action
(context);
     gtk_drag_get_data (widget, context,
target, time);
   }

  return TRUE;
}

static void
drag_data_received (GtkWidget        *widget,
                    GdkDragContext
*context,
                    gint             x,
                    gint             y,
                    GtkSelectionData
*selection_data,
                    guint            info,
                    guint            time)
{
  PrivateData *private_data =
GET_PRIVATE_DATA (widget);

  if (private_data->suggested_action)
   {
     private_data->suggested_action = 0;

     // We are getting this data due to a
request in drag_motion,
     // rather than due to a request in
drag_drop, so we are just
```

```
58                                          // supposed to call gdk_drag_status(),
59                                      not actually paste in
60                                          // the data.
61
62                                          str = gtk_selection_data_get_text
63                                      (selection_data);
                                            if (!data_is_acceptable (str))
                                              gdk_drag_status (context, 0, time);
                                            else
                                              gdk_drag_status (context,
                                                               private_data-
                                      >suggested_action,
                                                               time);
                                       }
                                      else
                                       {
                                         // accept the drop
                                       }
                                      }
```

Parameters

widget	the object which received the signal
context	the drag context
x	the x coordinate of the current cursor position
y	the y coordinate of the current cursor position
time	the timestamp of the motion event
user_data	user data set when the signal handler was connected.

Returns

whether the cursor position is in a drop zone

Flags: Run Last

The “draw” signal

```
gboolean
user_function (GtkWidget    *widget,
               CairoContext *cr,
               gpointer      user_data)
```

This signal is emitted when a widget is supposed to render itself. The `widget` 's top left corner must be painted at the origin of the passed in context and be sized to the values returned by `gtk_widget_get_allocated_width()` and `gtk_widget_get_allocated_height()`.

Signal handlers connected to this signal can modify the cairo context passed as `cr` in any way they like and don't need to restore it. The signal emission takes care of calling `cairo_save()` before and `cairo_restore()` after invoking the handler.

The signal handler will get a `cr` with a clip region already set to the widget's dirty region, i.e. to the area that needs repainting. Complicated widgets that want to avoid redrawing themselves completely can get the full

extents of the clip region with `gdk_cairo_get_clip_rectangle()`, or they can get a finer-grained representation of the dirty region with `cairo_copy_clip_rectangle_list()`.

Parameters

widget	the object which received the signal	
cr	the cairo context to draw to	
user_data	user data set when the signal handler was connected.	

Returns

`TRUE` to stop other handlers from being invoked for the event. `FALSE` to propagate the event further.

Flags: Run Last

Since: 3.0

The `"enter-notify-event"` signal

```
gboolean
user_function (GtkWidget *widget,
               GdkEvent  *event,
               gpointer   user_data)
```

The ::enter-notify-event will be emitted when the pointer enters the `widget` 's window.

To receive this signal, the GdkWindow associated to the widget needs to enable the GDK_ENTER_NOTIFY_MASK mask.

This signal will be sent to the grab widget if there is one.

Parameters

widget	the object which received the signal	
event	the GdkEventCrossing which triggered this signal.	[type Gdk.EventCrossing]
user_data	user data set when the signal handler was connected.	

Returns

`TRUE` to stop other handlers from being invoked for the event. `FALSE` to propagate the event further.

Flags: Run Last

The “event” signal

```
gboolean
user_function (GtkWidget *widget,
               GdkEvent  *event,
               gpointer   user_data)
```

The GTK+ main loop will emit three signals for each GDK event delivered to a widget: one generic ::event signal, another, more specific, signal that matches the type of event delivered (e.g. “key-press-event”) and finally a generic “event-after” signal.

Parameters

widget	the object which received the signal.
event	the GdkEvent which triggered this signal
user_data	user data set when the signal handler was connected.

Returns

TRUE to stop other handlers from being invoked for the event and to cancel the emission of the second specific ::event signal. FALSE to propagate the event further and to allow the emission of the second signal. The ::event-after signal is emitted regardless of the return value.

Flags: Run Last

The “event-after” signal

```
void
user_function (GtkWidget *widget,
               GdkEvent  *event,
               gpointer   user_data)
```

After the emission of the “event” signal and (optionally) the second more specific signal, ::event-after will be emitted regardless of the previous two signals handlers return values.

Parameters

widget	the object which received the signal.
event	the GdkEvent which triggered this signal
user_data	user data set when the signal handler was connected.

The “focus” signal

```
gboolean
user_function (GtkWidget       *widget,
               GtkDirectionType direction,
```

```
              gpointer         user_data)
```

Parameters

widget	the object which received the signal.
user_data	user data set when the signal handler was connected.

Returns

`TRUE` to stop other handlers from being invoked for the event. `FALSE` to propagate the event further.

Flags: Run Last

The `"focus-in-event"` signal

```
gboolean
user_function (GtkWidget *widget,
               GdkEvent  *event,
               gpointer   user_data)
```

The ::focus-in-event signal will be emitted when the keyboard focus enters the `widget` 's window.

To receive this signal, the GdkWindow associated to the widget needs to enable the GDK_FOCUS_CHANGE_MASK mask.

Parameters

widget	the object which received the signal
event	the GdkEventFocus which triggered this signal. [type Gdk.EventFocus]
user_data	user data set when the signal handler was connected.

Returns

`TRUE` to stop other handlers from being invoked for the event. `FALSE` to propagate the event further.

Flags: Run Last

The `"focus-out-event"` signal

```
gboolean
user_function (GtkWidget *widget,
               GdkEvent  *event,
               gpointer   user_data)
```

The ::focus-out-event signal will be emitted when the keyboard focus leaves the `widget` 's window.

To receive this signal, the GdkWindow associated to the widget needs to enable the GDK_FOCUS_CHANGE_MASK mask.

Parameters

widget	the object which received the signal	
event	the GdkEventFocus which triggered this signal.	[type Gdk.EventFocus]
user_data	user data set when the signal handler was connected.	

Returns

`TRUE` to stop other handlers from being invoked for the event. `FALSE` to propagate the event further.

Flags: Run Last

The `"grab-broken-event"` signal

```
gboolean
user_function (GtkWidget *widget,
               GdkEvent  *event,
               gpointer   user_data)
```

Emitted when a pointer or keyboard grab on a window belonging to `widget` gets broken.

On X11, this happens when the grab window becomes unviewable (i.e. it or one of its ancestors is unmapped), or if the same application grabs the pointer or keyboard again.

Parameters

widget	the object which received the signal	
event	the GdkEventGrabBroken event.	[type Gdk.EventGrabBroken]
user_data	user data set when the signal handler was connected.	

Returns

`TRUE` to stop other handlers from being invoked for the event. `FALSE` to propagate the event further.

Flags: Run Last

Since: 2.8

The `"grab-focus"` signal

```
void
user_function (GtkWidget *widget,
               gpointer   user_data)
```

Parameters

widget	the object which received the signal.
user_data	user data set when the signal handler was connected.

Flags: Action

The `"grab-notify"` signal

```
void
user_function (GtkWidget *widget,
               gboolean   was_grabbed,
               gpointer   user_data)
```

The ::grab-notify signal is emitted when a widget becomes shadowed by a GTK+ grab (not a pointer or keyboard grab) on another widget, or when it becomes unshadowed due to a grab being removed.

A widget is shadowed by a `gtk_grab_add()` when the topmost grab widget in the grab stack of its window group is not its ancestor.

Parameters

widget	the object which received the signal
was_grabbed	`FALSE` if the widget becomes shadowed, `TRUE` if it becomes unshadowed
user_data	user data set when the signal handler was connected.

Flags: Run First

The `"hide"` signal

```
void
user_function (GtkWidget *widget,
               gpointer   user_data)
```

The ::hide signal is emitted when `widget` is hidden, for example with `gtk_widget_hide()`.

Parameters

widget	the object which received the signal.
user_data	user data set when the signal handler was connected.

Flags: Run First

The `"hierarchy-changed"` signal

```
void
```

```
user_function (GtkWidget *widget,
               GtkWidget *previous_toplevel,
               gpointer   user_data)
```

The ::hierarchy-changed signal is emitted when the anchored state of a widget changes. A widget is “anchored” when its toplevel ancestor is a GtkWindow. This signal is emitted when a widget changes from un-anchored to anchored or vice-versa.

Parameters

widget	the object on which the signal is emitted	
previous_toplevel	the previous toplevel ancestor, or `NULL` if the widget was previously unanchored.	[allow-none]
user_data	user data set when the signal handler was connected.	

Flags: Run Last

The `“key-press-event”` signal

```
gboolean
user_function (GtkWidget *widget,
               GdkEvent  *event,
               gpointer   user_data)
```

The ::key-press-event signal is emitted when a key is pressed. The signal emission will reoccur at the key-repeat rate when the key is kept pressed.

To receive this signal, the GdkWindow associated to the widget needs to enable the GDK_KEY_PRESS_MASK mask.

This signal will be sent to the grab widget if there is one.

Parameters

widget	the object which received the signal	
event	the GdkEventKey which triggered this signal.	[type Gdk.EventKey]
user_data	user data set when the signal handler was connected.	

Returns

`TRUE` to stop other handlers from being invoked for the event. `FALSE` to propagate the event further.

Flags: Run Last

The “key-release-event” signal

```
gboolean
user_function (GtkWidget *widget,
               GdkEvent  *event,
               gpointer   user_data)
```

The ::key-release-event signal is emitted when a key is released.

To receive this signal, the GdkWindow associated to the widget needs to enable the GDK_KEY_RELEASE_MASK mask.

This signal will be sent to the grab widget if there is one.

Parameters

widget	the object which received the signal	
event	the GdkEventKey which triggered this signal.	[type Gdk.EventKey]
user_data	user data set when the signal handler was connected.	

Returns

TRUE to stop other handlers from being invoked for the event. FALSE to propagate the event further.

Flags: Run Last

The “keynav-failed” signal

```
gboolean
user_function (GtkWidget       *widget,
               GtkDirectionType direction,
               gpointer         user_data)
```

Gets emitted if keyboard navigation fails. See gtk_widget_keynav_failed() for details.

Parameters

widget	the object which received the signal
direction	the direction of movement
user_data	user data set when the signal handler was connected.

Returns

TRUE if stopping keyboard navigation is fine, FALSE if the emitting widget should try to handle the keyboard navigation attempt in its parent container(s).

Flags: Run Last

Since: 2.12

The “leave-notify-event” signal

```
gboolean
user_function (GtkWidget *widget,
               GdkEvent  *event,
               gpointer   user_data)
```

The ::leave-notify-event will be emitted when the pointer leaves the `widget` 's window.

To receive this signal, the GdkWindow associated to the widget needs to enable the GDK_LEAVE_NOTIFY_MASK mask.

This signal will be sent to the grab widget if there is one.

Parameters

widget	the object which received the signal	
event	the GdkEventCrossing which triggered this signal.	[type Gdk.EventCrossing]
user_data	user data set when the signal handler was connected.	

Returns

`TRUE` to stop other handlers from being invoked for the event. `FALSE` to propagate the event further.

Flags: Run Last

The “map” signal

```
void
user_function (GtkWidget *widget,
               gpointer   user_data)
```

The ::map signal is emitted when `widget` is going to be mapped, that is when the widget is visible (which is controlled with `gtk_widget_set_visible()`) and all its parents up to the toplevel widget are also visible. Once the map has occurred, “map-event” will be emitted.

The ::map signal can be used to determine whether a widget will be drawn, for instance it can resume an animation that was stopped during the emission of “unmap”.

Parameters

widget	the object which received the signal.
user_data	user data set when the signal handler was connected.

Flags: Run First

The “map-event” signal

```
gboolean
user_function (GtkWidget *widget,
               GdkEvent  *event,
               gpointer   user_data)
```

The ::map-event signal will be emitted when the `widget` 's window is mapped. A window is mapped when it becomes visible on the screen.

To receive this signal, the GdkWindow associated to the widget needs to enable the GDK_STRUCTURE_MASK mask. GDK will enable this mask automatically for all new windows.

Parameters

widget	the object which received the signal	
event	the GdkEventAny which triggered this signal.	[type Gdk.EventAny]
user_data	user data set when the signal handler was connected.	

Returns

`TRUE` to stop other handlers from being invoked for the event. `FALSE` to propagate the event further.

Flags: Run Last

The “mnemonic-activate” signal

```
gboolean
user_function (GtkWidget *widget,
               gboolean   group_cycling,
               gpointer   user_data)
```

The default handler for this signal activates `widget` if `group_cycling` is `FALSE`, or just makes `widget` grab focus if `group_cycling` is `TRUE`.

Parameters

widget	the object which received the signal.
group_cycling	`TRUE` if there are other widgets with the same mnemonic
user_data	user data set when the signal handler was connected.

Returns

`TRUE` to stop other handlers from being invoked for the event. `FALSE` to propagate the event further.

Flags: Run Last

The “motion-notify-event” signal

```
gboolean
user_function (GtkWidget *widget,
               GdkEvent  *event,
               gpointer   user_data)
```

The ::motion-notify-event signal is emitted when the pointer moves over the widget's GdkWindow.

To receive this signal, the GdkWindow associated to the widget needs to enable the GDK_POINTER_MOTION_MASK mask.

This signal will be sent to the grab widget if there is one.

Parameters

widget	the object which received the signal.	
event	the GdkEventMotion which triggered this signal.	[type Gdk.EventMotion]
user_data	user data set when the signal handler was connected.	

Returns

TRUE to stop other handlers from being invoked for the event. FALSE to propagate the event further.

Flags: Run Last

The “move-focus” signal

```
void
user_function (GtkWidget       *widget,
               GtkDirectionType direction,
               gpointer         user_data)
```

Parameters

widget	the object which received the signal.
user_data	user data set when the signal handler was connected.

Flags: Action

The “parent-set” signal

```
void
user_function (GtkWidget *widget,
               GtkWidget *old_parent,
               gpointer   user_data)
```

The ::parent-set signal is emitted when a new parent has been set on a widget.

Parameters

widget	the object on which the signal is emitted	
old_parent	the previous parent, or `NULL` if the widget just got its initial parent.	[allow-none]
user_data	user data set when the signal handler was connected.	

Flags: Run First

The `"popup-menu"` signal

```
gboolean
user_function (GtkWidget *widget,
               gpointer   user_data)
```

This signal gets emitted whenever a widget should pop up a context menu. This usually happens through the standard key binding mechanism; by pressing a certain key while a widget is focused, the user can cause the widget to pop up a menu. For example, the GtkEntry widget creates a menu with clipboard commands. See the Popup Menu Migration Checklist for an example of how to use this signal.

Parameters

widget	the object which received the signal
user_data	user data set when the signal handler was connected.

Returns

`TRUE` if a menu was activated

Flags: Action

The `"property-notify-event"` signal

```
gboolean
user_function (GtkWidget *widget,
               GdkEvent  *event,
               gpointer   user_data)
```

The ::property-notify-event signal will be emitted when a property on the `widget` 's window has been changed or deleted.

To receive this signal, the GdkWindow associated to the widget needs to enable the GDK_PROPERTY_CHANGE_MASK mask.

Parameters

widget	the object which received the signal	
event	the GdkEventProperty which triggered this signal.	[type Gdk.EventProperty]
user_data	user data set when the signal handler was connected.	

Returns

`TRUE` to stop other handlers from being invoked for the event. `FALSE` to propagate the event further.

Flags: Run Last

The `"proximity-in-event"` signal

```
gboolean
user_function (GtkWidget *widget,
               GdkEvent  *event,
               gpointer   user_data)
```

To receive this signal the GdkWindow associated to the widget needs to enable the GDK_PROXIMITY_IN_MASK mask.

This signal will be sent to the grab widget if there is one.

Parameters

widget	the object which received the signal	
event	the GdkEventProximity which triggered this signal.	[type Gdk.EventProximity]
user_data	user data set when the signal handler was connected.	

Returns

`TRUE` to stop other handlers from being invoked for the event. `FALSE` to propagate the event further.

Flags: Run Last

The `"proximity-out-event"` signal

```
gboolean
user_function (GtkWidget *widget,
               GdkEvent  *event,
               gpointer   user_data)
```

To receive this signal the GdkWindow associated to the widget needs to enable the GDK_PROXIMITY_OUT_MASK mask.

This signal will be sent to the grab widget if there is one.

Parameters

widget	the object which received the signal	
event	the GdkEventProximity which triggered this signal.	[type Gdk.EventProximity]
user_data	user data set when the signal handler was connected.	

Returns

`TRUE` to stop other handlers from being invoked for the event. `FALSE` to propagate the event further.

Flags: Run Last

The `"query-tooltip"` signal

```
gboolean
user_function (GtkWidget  *widget,
               gint        x,
               gint        y,
               gboolean    keyboard_mode,
               GtkTooltip *tooltip,
               gpointer    user_data)
```

Emitted when "has-tooltip" is `TRUE` and the hover timeout has expired with the cursor hovering "above" `widget` ; or emitted when `widget` got focus in keyboard mode.

Using the given coordinates, the signal handler should determine whether a tooltip should be shown for `widget` . If this is the case `TRUE` should be returned, `FALSE` otherwise. Note that if `keyboard_mode` is `TRUE`, the values of `x` and `y` are undefined and should not be used.

The signal handler is free to manipulate `tooltip` with the therefore destined function calls.

Parameters

widget	the object which received the signal
x	the x coordinate of the cursor position where the request has been emitted, relative to `widget` 's left side
y	the y coordinate of the cursor position where the request has been emitted, relative to `widget` 's top
keyboard_mode	`TRUE` if the tooltip was triggered using the keyboard
tooltip	a GtkTooltip
user_data	user data set when the signal handler was connected.

Returns

TRUE if tooltip should be shown right now, FALSE otherwise.

Flags: Run Last

Since: 2.12

The "realize" signal

```
void
user_function (GtkWidget *widget,
               gpointer   user_data)
```

The ::realize signal is emitted when widget is associated with a GdkWindow, which means that gtk_widget_realize() has been called or the widget has been mapped (that is, it is going to be drawn).

Parameters

widget	the object which received the signal.	
user_data	user data set when the signal handler was connected.	

Flags: Run First

The "screen-changed" signal

```
void
user_function (GtkWidget *widget,
               GdkScreen *previous_screen,
               gpointer   user_data)
```

The ::screen-changed signal gets emitted when the screen of a widget has changed.

Parameters

widget	the object on which the signal is emitted	
previous_screen	the previous screen, or NULL if the widget was not associated with a screen before.	[allow-none]
user_data	user data set when the signal handler was connected.	

Flags: Run Last

The "scroll-event" signal

```
gboolean
user_function (GtkWidget *widget,
               GdkEvent  *event,
```

```
                gpointer   user_data)
```

The ::scroll-event signal is emitted when a button in the 4 to 7 range is pressed. Wheel mice are usually configured to generate button press events for buttons 4 and 5 when the wheel is turned.

To receive this signal, the GdkWindow associated to the widget needs to enable the GDK_SCROLL_MASK mask.

This signal will be sent to the grab widget if there is one.

Parameters

widget	the object which received the signal.	
event	the GdkEventScroll which triggered this signal.	[type Gdk.EventScroll]
user_data	user data set when the signal handler was connected.	

Returns

`TRUE` to stop other handlers from being invoked for the event. `FALSE` to propagate the event further.

Flags: Run Last

The `"selection-clear-event"` signal

```
gboolean
user_function (GtkWidget *widget,
               GdkEvent  *event,
               gpointer   user_data)
```

The ::selection-clear-event signal will be emitted when the the `widget` 's window has lost ownership of a selection.

Parameters

widget	the object which received the signal	
event	the GdkEventSelection which triggered this signal.	[type Gdk.EventSelection]
user_data	user data set when the signal handler was connected.	

Returns

`TRUE` to stop other handlers from being invoked for the event. `FALSE` to propagate the event further.

Flags: Run Last

The `"selection-get"` signal

```
void
user_function (GtkWidget        *widget,
               GtkSelectionData *data,
               guint             info,
               guint             time,
               gpointer          user_data)
```

Parameters

widget	the object which received the signal.
user_data	user data set when the signal handler was connected.

Flags: Run Last

The `"selection-notify-event"` signal

```
gboolean
user_function (GtkWidget *widget,
               GdkEvent  *event,
               gpointer   user_data)
```

Parameters

widget	the object which received the signal.	
event	.	[type Gdk.EventSelection]
user_data	user data set when the signal handler was connected.	

Returns

`TRUE` to stop other handlers from being invoked for the event. `FALSE` to propagate the event further.

Flags: Run Last

The `"selection-received"` signal

```
void
user_function (GtkWidget        *widget,
               GtkSelectionData *data,
               guint             time,
               gpointer          user_data)
```

Parameters

widget	the object which received the signal.
user_data	user data set when the signal handler was connected.

Flags: Run Last

The “selection-request-event” signal

```
gboolean
user_function (GtkWidget *widget,
               GdkEvent  *event,
               gpointer   user_data)
```

The ::selection-request-event signal will be emitted when another client requests ownership of the selection owned by the `widget` 's window.

Parameters

widget	the object which received the signal	
event	the GdkEventSelection which triggered this signal.	[type Gdk.EventSelection]
user_data	user data set when the signal handler was connected.	

Returns

`TRUE` to stop other handlers from being invoked for the event. `FALSE` to propagate the event further.

Flags: Run Last

The “show” signal

```
void
user_function (GtkWidget *widget,
               gpointer   user_data)
```

The ::show signal is emitted when `widget` is shown, for example with `gtk_widget_show()`.

Parameters

widget	the object which received the signal.
user_data	user data set when the signal handler was connected.

Flags: Run First

The “show-help” signal

```
gboolean
user_function (GtkWidget        *widget,
               GtkWidgetHelpType help_type,
               gpointer          user_data)
```

Parameters

widget	the object which received the signal.
user_data	user data set when the signal handler was connected.

Returns

`TRUE` to stop other handlers from being invoked for the event. `FALSE` to propagate the event further.

Flags: Action

The `"size-allocate"` signal

```
void
user_function (GtkWidget    *widget,
               GdkRectangle *allocation,
               gpointer      user_data)
```

Parameters

widget	the object which received the signal.
allocation	the region which has been allocated [type Gtk.Allocation] to the widget.
user_data	user data set when the signal handler was connected.

Flags: Run First

The `"state-changed"` signal

```
void
user_function (GtkWidget   *widget,
               GtkStateType state,
               gpointer     user_data)
```

The ::state-changed signal is emitted when the widget state changes. See `gtk_widget_get_state()`.

`GtkWidget::state-changed` has been deprecated since version 3.0 and should not be used in newly-written code.

Use "state-flags-changed" instead.

Parameters

widget	the object which received the signal.
state	the previous state
user_data	user data set when the signal handler was connected.

Flags: Run First

The “state-flags-changed” signal

```
void
user_function (GtkWidget    *widget,
               GtkStateFlags flags,
               gpointer      user_data)
```

The ::state-flags-changed signal is emitted when the widget state changes, see gtk_widget_get_state_flags().

Parameters

widget	the object which received the signal.	
flags	The previous state flags.	
user_data	user data set when the signal handler was connected.	

Flags: Run First

Since: 3.0

The “style-set” signal

```
void
user_function (GtkWidget *widget,
               GtkStyle  *previous_style,
               gpointer   user_data)
```

The ::style-set signal is emitted when a new style has been set on a widget. Note that style-modifying functions like gtk_widget_modify_base() also cause this signal to be emitted.

Note that this signal is emitted for changes to the deprecated GtkStyle. To track changes to the GtkStyleContext associated with a widget, use the “style-updated” signal.

GtkWidget::style-set has been deprecated since version 3.0 and should not be used in newly-written code.

Use the “style-updated” signal

Parameters

widget	the object on which the signal is emitted	
previous_style	the previous style, or NULL if the widget just got its initial style.	[allow-none]
user_data	user data set when the signal handler was connected.	

Flags: Run First

The “style-updated” signal

```
void
user_function (GtkWidget *widget,
               gpointer   user_data)
```

The ::style-updated signal is a convenience signal that is emitted when the “changed” signal is emitted on the `widget` 's associated GtkStyleContext as returned by `gtk_widget_get_style_context()`.

Note that style-modifying functions like `gtk_widget_override_color()` also cause this signal to be emitted.

Parameters

widget	the object on which the signal is emitted
user_data	user data set when the signal handler was connected.

Flags: Run First

Since: 3.0

The “touch-event” signal

```
gboolean
user_function (GtkWidget *widget,
               GdkEvent  *arg1,
               gpointer   user_data)
```

Flags: Run Last

The “unmap” signal

```
void
user_function (GtkWidget *widget,
               gpointer   user_data)
```

The ::unmap signal is emitted when `widget` is going to be unmapped, which means that either it or any of its parents up to the toplevel widget have been set as hidden.

As ::unmap indicates that a widget will not be shown any longer, it can be used to, for example, stop an animation on the widget.

Parameters

widget	the object which received the signal.
user_data	user data set when the signal handler was connected.

Flags: Run First

The “unmap-event” signal

```
gboolean
user_function (GtkWidget *widget,
               GdkEvent  *event,
               gpointer   user_data)
```

The ::unmap-event signal will be emitted when the `widget` 's window is unmapped. A window is unmapped when it becomes invisible on the screen.

To receive this signal, the GdkWindow associated to the widget needs to enable the GDK_STRUCTURE_MASK mask. GDK will enable this mask automatically for all new windows.

Parameters

widget	the object which received the signal	
event	the GdkEventAny which triggered this signal.	[type Gdk.EventAny]
user_data	user data set when the signal handler was connected.	

Returns

`TRUE` to stop other handlers from being invoked for the event. `FALSE` to propagate the event further.

Flags: Run Last

The “unrealize” signal

```
void
user_function (GtkWidget *widget,
               gpointer   user_data)
```

The ::unrealize signal is emitted when the GdkWindow associated with `widget` is destroyed, which means that `gtk_widget_unrealize()` has been called or the widget has been unmapped (that is, it is going to be hidden).

Parameters

widget	the object which received the signal.
user_data	user data set when the signal handler was connected.

Flags: Run Last

The “visibility-notify-event” signal

```
gboolean
user_function (GtkWidget *widget,
               GdkEvent  *event,
               gpointer   user_data)
```

The ::visibility-notify-event will be emitted when the `widget` 's window is obscured or unobscured.

To receive this signal the GdkWindow associated to the widget needs to enable the GDK_VISIBILITY_NOTIFY_MASK mask.

`GtkWidget::visibility-notify-event` has been deprecated since version 3.12 and should not be used in newly-written code.

Modern composited windowing systems with pervasive transparency make it impossible to track the visibility of a window reliably, so this signal can not be guaranteed to provide useful information.

Parameters

widget	the object which received the signal	
event	the GdkEventVisibility which triggered this signal.	[type Gdk.EventVisibility]
user_data	user data set when the signal handler was connected.	

Returns

`TRUE` to stop other handlers from being invoked for the event. `FALSE` to propagate the event further.

Flags: Run Last

The `"window-state-event"` signal

```
gboolean
user_function (GtkWidget *widget,
               GdkEvent  *event,
               gpointer   user_data)
```

The ::window-state-event will be emitted when the state of the toplevel window associated to the `widget` changes.

To receive this signal the GdkWindow associated to the widget needs to enable the GDK_STRUCTURE_MASK mask. GDK will enable this mask automatically for all new windows.

Parameters

widget	the object which received the signal	
event	the GdkEventWindowState which triggered this signal.	[type Gdk.EventWindowState]
user_data	user data set when the signal handler was connected.	

Returns

`TRUE` to stop other handlers from being invoked for the event. `FALSE` to propagate the event further.

Flags: Run Last

GtkContainer

GtkContainer — Base class for widgets which contain other widgets

Functions

#define	GTK_IS_RESIZE_CONTAINER()
#define	GTK_CONTAINER_WARN_INVALID_CHILD_PROPERTY_ID()
void	gtk_container_add ()
void	gtk_container_remove ()
void	gtk_container_add_with_properties ()
GtkResizeMode	gtk_container_get_resize_mode ()
void	gtk_container_set_resize_mode ()
void	gtk_container_check_resize ()
void	gtk_container_foreach ()
GList *	gtk_container_get_children ()
GtkWidgetPath *	gtk_container_get_path_for_child ()
void	gtk_container_set_reallocate_redraws ()
GtkWidget *	gtk_container_get_focus_child ()
void	gtk_container_set_focus_child ()
GtkAdjustment *	gtk_container_get_focus_vadjustment ()
void	gtk_container_set_focus_vadjustment ()
GtkAdjustment *	gtk_container_get_focus_hadjustment ()
void	gtk_container_set_focus_hadjustment ()
void	gtk_container_resize_children ()
GType	gtk_container_child_type ()
void	gtk_container_child_get ()
void	gtk_container_child_set ()
void	gtk_container_child_get_property ()
void	gtk_container_child_set_property ()
void	gtk_container_child_get_valist ()
void	gtk_container_child_set_valist ()
void	gtk_container_child_notify ()
void	gtk_container_child_notify_by_pspec ()
void	gtk_container_forall ()
guint	gtk_container_get_border_width ()
void	gtk_container_set_border_width ()
void	gtk_container_propagate_draw ()
gboolean	gtk_container_get_focus_chain ()
void	gtk_container_set_focus_chain ()
void	gtk_container_unset_focus_chain ()
GParamSpec *	gtk_container_class_find_child_property ()
void	gtk_container_class_install_child_property ()
void	gtk_container_class_install_child_properties ()
GParamSpec **	gtk_container_class_list_child_properties ()
void	gtk_container_class_handle_border_width ()

Properties

guint	border-width	Read / Write
GtkWidget *	child	Write
GtkResizeMode	resize-mode	Read / Write

Signals

void	add	Run First
void	check-resize	Run Last
void	remove	Run First
void	set-focus-child	Run First

Types and Values

struct	GtkContainer
struct	GtkContainerClass
enum	GtkResizeMode

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkContainer
            ├── GtkBin
            ├── GtkBox
            ├── GtkFixed
            ├── GtkFlowBox
            ├── GtkGrid
            ├── GtkHeaderBar
            ├── GtkPaned
            ├── GtkIconView
            ├── GtkLayout
            ├── GtkListBox
            ├── GtkMenuShell
            ├── GtkNotebook
            ├── GtkSocket
            ├── GtkStack
            ├── GtkTable
            ├── GtkTextView
            ├── GtkToolbar
            ├── GtkToolItemGroup
            ├── GtkToolPalette
            └── GtkTreeView
```

Implemented Interfaces

GtkContainer implements AtkImplementorIface and GtkBuildable.

Includes

```
#include <gtk/gtk.h>
```

Description

A GTK+ user interface is constructed by nesting widgets inside widgets. Container widgets are the inner nodes in the resulting tree of widgets: they contain other widgets. So, for example, you might have a GtkWindow containing a GtkFrame containing a GtkLabel. If you wanted an image instead of a textual label inside the frame, you might replace the GtkLabel widget with a GtkImage widget.

There are two major kinds of container widgets in GTK+. Both are subclasses of the abstract GtkContainer base class.

The first type of container widget has a single child widget and derives from GtkBin. These containers are decorators, which add some kind of functionality to the child. For example, a GtkButton makes its child into a clickable button; a GtkFrame draws a frame around its child and a GtkWindow places its child widget inside a top-level window.

The second type of container can have more than one child; its purpose is to manage layout. This means that these containers assign sizes and positions to their children. For example, a GtkHBox arranges its children in a horizontal row, and a GtkGrid arranges the widgets it contains in a two-dimensional grid.

For implementations of GtkContainer the virtual method `GtkContainerClass.forall()` is always required, since it's used for drawing and other internal operations on the children. If the GtkContainer implementation expect to have non internal children it's needed to implement both `GtkContainerClass.add()` and `GtkContainerClass.remove()`. If the GtkContainer implementation has internal children, they should be added with `gtk_widget_set_parent()` on `init()` and removed with `gtk_widget_unparent()` in the `GtkWidgetClass.destroy()` implementation. See more about implementing custom widgets at https://wiki.gnome.org/HowDoI/CustomWidgets

Height for width geometry management

GTK+ uses a height-for-width (and width-for-height) geometry management system. Height-for-width means that a widget can change how much vertical space it needs, depending on the amount of horizontal space that it is given (and similar for width-for-height).

There are some things to keep in mind when implementing container widgets that make use of GTK+'s height for width geometry management system. First, it's important to note that a container must prioritize one of its dimensions, that is to say that a widget or container can only have a GtkSizeRequestMode that is `GTK_SIZE_REQUEST_HEIGHT_FOR_WIDTH` or `GTK_SIZE_REQUEST_WIDTH_FOR_HEIGHT`. However, every widget and container must be able to respond to the APIs for both dimensions, i.e. even if a widget has a request mode that is height-for-width, it is possible that its parent will request its sizes using the width-for-height APIs.

To ensure that everything works properly, here are some guidelines to follow when implementing height-for-width (or width-for-height) containers.

Each request mode involves 2 virtual methods. Height-for-width apis run through `gtk_widget_get_preferred_width()` and then through `gtk_widget_get_preferred_height_for_width()`. When handling requests in the opposite GtkSizeRequestMode it is important that every widget request at least enough space to display all of its content at all times.

When `gtk_widget_get_preferred_height()` is called on a container that is height-for-width, the container must return the height for its minimum width. This is easily achieved by simply calling the reverse apis implemented for itself as follows:

```
1                                               static void
2                                               foo_container_get_preferred_height (GtkWidget
```

```
3                                                   *widget,
4                                                                       gint
5                                                   *min_height,
6                                                                       gint
7                                                   *nat_height)
8                                                   {
9                                                      if (i_am_in_height_for_width_mode)
10                                                       {
11                                                       gint min_width;
12
13                                                       GTK_WIDGET_GET_CLASS (widget)-
14                                                  >get_preferred_width (widget,
15
16                                                  &min_width,
17
18                                                  NULL);
19                                                       GTK_WIDGET_GET_CLASS (widget)-
20                                                  >get_preferred_height_for_width
21
22                                                  (widget,
23
24                                                  min_width,
25
26                                                  min_height,
27
                                                    nat_height);
                                                       }
                                                     else
                                                       {
                                                       ... many containers support both
                                                  request modes, execute the
                                                       real width-for-height request here by
                                                  returning the
                                                       collective heights of all widgets that
                                                  are stacked
                                                       vertically (or whatever is appropriate
                                                  for this container)
                                                       ...
                                                       }
                                                  }
```

Similarly, when `gtk_widget_get_preferred_width_for_height()` is called for a container or widget that is height-for-width, it then only needs to return the base minimum width like so:

```
1                                                   static void
2                                                   foo_container_get_preferred_width_for_height
3                                                   (GtkWidget *widget,
4
5                                                   gint for_height,
6
7                                                   gint *min_width,
8
9                                                   gint *nat_width)
10                                                  {
11                                                     if (i_am_in_height_for_width_mode)
12                                                       {
13                                                       GTK_WIDGET_GET_CLASS (widget)-
14                                                  >get_preferred_width (widget,
15
16                                                  min_width,
17
18                                                  nat_width);
19                                                       }
                                                     else
```

```
    {
      ... execute the real width-for-height
request here based on
       the required width of the children
collectively if the
       container were to be allocated the
said height ...
    }
}
```

Height for width requests are generally implemented in terms of a virtual allocation of widgets in the input orientation. Assuming an height-for-width request mode, a container would implement the `get_preferred_height_for_width()` virtual function by first calling `gtk_widget_get_preferred_width()` for each of its children.

For each potential group of children that are lined up horizontally, the values returned by `gtk_widget_get_preferred_width()` should be collected in an array of GtkRequestedSize structures. Any child spacing should be removed from the input `for_width` and then the collective size should be allocated using the `gtk_distribute_natural_allocation()` convenience function.

The container will then move on to request the preferred height for each child by using `gtk_widget_get_preferred_height_for_width()` and using the sizes stored in the GtkRequestedSize array.

To allocate a height-for-width container, it's again important to consider that a container must prioritize one dimension over the other. So if a container is a height-for-width container it must first allocate all widgets horizontally using a GtkRequestedSize array and `gtk_distribute_natural_allocation()` and then add any extra space (if and where appropriate) for the widget to expand.

After adding all the expand space, the container assumes it was allocated sufficient height to fit all of its content. At this time, the container must use the total horizontal sizes of each widget to request the height-for-width of each of its children and store the requests in a GtkRequestedSize array for any widgets that stack vertically (for tabular containers this can be generalized into the heights and widths of rows and columns). The vertical space must then again be distributed using `gtk_distribute_natural_allocation()` while this time considering the allocated height of the widget minus any vertical spacing that the container adds. Then vertical expand space should be added where appropriate and available and the container should go on to actually allocating the child widgets.

See GtkWidget's geometry management section to learn more about implementing height-for-width geometry management for widgets.

Child properties

GtkContainer introduces child properties. These are object properties that are not specific to either the container or the contained widget, but rather to their relation. Typical examples of child properties are the position or pack-type of a widget which is contained in a GtkBox.

Use `gtk_container_class_install_child_property()` to install child properties for a container class and `gtk_container_class_find_child_property()` or `gtk_container_class_list_child_properties()` to get information about existing child properties.

To set the value of a child property, use `gtk_container_child_set_property()`, `gtk_container_child_set()` or `gtk_container_child_set_valist()`. To obtain the value of a child property, use `gtk_container_child_get_property()`, `gtk_container_child_get()` or `gtk_container_child_get_valist()`. To emit notification about child property changes, use `gtk_widget_child_notify()`.

GtkContainer as GtkBuildable

The GtkContainer implementation of the GtkBuildable interface supports a <packing> element for children, which can contain multiple <property> elements that specify child properties for the child.

Since 2.16, child properties can also be marked as translatable using the same “translatable”, “comments” and “context” attributes that are used for regular properties.

Since 3.16, containers can have a <focus-chain> element containing multiple <widget> elements, one for each child that should be added to the focus chain. The ”name” attribute gives the id of the widget.

An example of these properties in UI definitions:

```
<object class="GtkBox">
  <child>
    <object class="GtkEntry" id="entry1"/>
    <packing>
      <property
name="pack-type">start</property>
    </packing>
  </child>
  <child>
    <object class="GtkEntry" id="entry2"/>
  </child>
  <focus-chain>
    <widget name="entry1"/>
    <widget name="entry2"/>
  </focus-chain>
</object>
```

Functions

GTK_IS_RESIZE_CONTAINER()

```
#define             GTK_IS_RESIZE_CONTAINER(widget)
```

GTK_CONTAINER_WARN_INVALID_CHILD_PROPERTY_ID()

```
#define             GTK_CONTAINER_WARN_INVALID_CHILD_PROPERTY_ID(object, property_id,
pspec)
```

This macro should be used to emit a standard warning about unexpected properties in `set_child_property()` and `get_child_property()` implementations.

Parameters

object	the GObject on which `set_child_property()` or `get_child_property()` was called
property_id	the numeric id of the property

pspec	the GParamSpec of the property

gtk_container_add ()

```
void
gtk_container_add (GtkContainer *container,
                   GtkWidget *widget);
```

Adds `widget` to `container` . Typically used for simple containers such as GtkWindow, GtkFrame, or GtkButton; for more complicated layout containers such as GtkBox or GtkGrid, this function will pick default packing parameters that may not be correct. So consider functions such as `gtk_box_pack_start()` and `gtk_grid_attach()` as an alternative to `gtk_container_add()` in those cases. A widget may be added to only one container at a time; you can't place the same widget inside two different containers.

Note that some containers, such as GtkScrolledWindow or GtkListBox, may add intermediate children between the added widget and the container.

Parameters

container	a GtkContainer
widget	a widget to be placed inside `container`

gtk_container_remove ()

```
void
gtk_container_remove (GtkContainer *container,
                      GtkWidget *widget);
```

Removes `widget` from `container` . `widget` must be inside `container` . Note that `container` will own a reference to `widget` , and that this may be the last reference held; so removing a widget from its container can destroy that widget. If you want to use `widget` again, you need to add a reference to it before removing it from a container, using `g_object_ref()`. If you don't want to use `widget` again it's usually more efficient to simply destroy it directly using `gtk_widget_destroy()` since this will remove it from the container and help break any circular reference count cycles.

Parameters

container	a GtkContainer
widget	a current child of `container`

gtk_container_add_with_properties ()

```
void
gtk_container_add_with_properties (GtkContainer *container,
                                   GtkWidget *widget,
                                   const gchar *first_prop_name,
                                   ...);
```

Adds `widget` to `container` , setting child properties at the same time. See `gtk_container_add()` and `gtk_container_child_set()` for more details.

Parameters

container	a GtkContainer
widget	a widget to be placed inside `container`
first_prop_name	the name of the first child property to set
...	a `NULL`-terminated list of property names and values, starting with `first_prop_name`

---

gtk_container_get_resize_mode ()

```
GtkResizeMode
gtk_container_get_resize_mode (GtkContainer *container);
```

`gtk_container_get_resize_mode` has been deprecated since version 3.12 and should not be used in newly-written code.

Resize modes are deprecated. They aren’t necessary anymore since frame clocks and might introduce obscure bugs if used.

Returns the resize mode for the container. See `gtk_container_set_resize_mode()`.

Parameters

container	a GtkContainer

Returns

the current resize mode

---

gtk_container_set_resize_mode ()

```
void
gtk_container_set_resize_mode (GtkContainer *container,
                               GtkResizeMode resize_mode);
```

`gtk_container_set_resize_mode` has been deprecated since version 3.12 and should not be used in newly-written code.

Resize modes are deprecated. They aren’t necessary anymore since frame clocks and might introduce obscure bugs if used.

Sets the resize mode for the container.

The resize mode of a container determines whether a resize request will be passed to the container’s parent, queued for later execution or executed immediately.

Parameters

container	a GtkContainer
resize_mode	the new resize mode

---

gtk_container_check_resize ()

```
void
gtk_container_check_resize (GtkContainer *container);
```

---

gtk_container_foreach ()

```
void
gtk_container_foreach (GtkContainer *container,
                       GtkCallback callback,
                       gpointer callback_data);
```

Invokes `callback` on each non-internal child of `container` . See `gtk_container_forall()` for details on what constitutes an “internal” child. For all practical purposes, this function should iterate over precisely those child widgets that were added to the container by the application with explicit `add()` calls.

It is permissible to remove the child from the `callback` handler.

Most applications should use `gtk_container_foreach()`, rather than `gtk_container_forall()`.

Parameters

container	a GtkContainer	
callback	a callback.	[scope call]
callback_data	callback user data	

---

gtk_container_get_children ()

```
GList *
gtk_container_get_children (GtkContainer *container);
```

Returns the container’s non-internal children. See `gtk_container_forall()` for details on what constitutes an "internal" child.

Parameters

container	a GtkContainer

Returns

a newly-allocated list of the container’s non-internal children.

[element-type GtkWidget][transfer container]

---

gtk_container_get_path_for_child ()

```
GtkWidgetPath *
gtk_container_get_path_for_child (GtkContainer *container,
                                  GtkWidget *child);
```

Returns a newly created widget path representing all the widget hierarchy from the toplevel down to and including `child` .

Parameters

container	a GtkContainer
child	a child of `container`

Returns

A newly created GtkWidgetPath

---

gtk_container_set_reallocate_redraws ()

```
void
gtk_container_set_reallocate_redraws (GtkContainer *container,
                                      gboolean needs_redraws);
```

`gtk_container_set_reallocate_redraws` has been deprecated since version 3.14 and should not be used in newly-written code.

Call `gtk_widget_queue_draw()` in your size_allocate handler.

Sets the `reallocate_redraws` flag of the container to the given value.

Containers requesting reallocation redraws get automatically redrawn if any of their children changed allocation.

Parameters

container	a GtkContainer
needs_redraws	the new value for the container's `reallocate_redraws` flag

---

gtk_container_get_focus_child ()

```
GtkWidget *
gtk_container_get_focus_child (GtkContainer *container);
```

Returns the current focus child widget inside `container` . This is not the currently focused widget. That can be

obtained by calling `gtk_window_get_focus()`.

Parameters

container	a GtkContainer	

Returns

The child widget which will receive the focus inside `container` when the `container` is focused, or `NULL` if none is set.

[nullable][transfer none]

Since: 2.14

---

gtk_container_set_focus_child ()

```
void
gtk_container_set_focus_child (GtkContainer *container,
                               GtkWidget *child);
```

Sets, or unsets if `child` is `NULL`, the focused child of `container` .

This function emits the GtkContainer::set_focus_child signal of `container` . Implementations of GtkContainer can override the default behaviour by overriding the class closure of this signal.

This is function is mostly meant to be used by widgets. Applications can use `gtk_widget_grab_focus()` to manually set the focus to a specific widget.

Parameters

container	a GtkContainer	
child	a GtkWidget, or `NULL`.	[allow-none]

---

gtk_container_get_focus_vadjustment ()

```
GtkAdjustment *
gtk_container_get_focus_vadjustment (GtkContainer *container);
```

Retrieves the vertical focus adjustment for the container. See `gtk_container_set_focus_vadjustment()`.

Parameters

container	a GtkContainer	

Returns

the vertical focus adjustment, or `NULL` if none has been set.

[nullable][transfer none]

---

gtk_container_set_focus_vadjustment ()

```
void
gtk_container_set_focus_vadjustment (GtkContainer *container,
                                     GtkAdjustment *adjustment);
```

Hooks up an adjustment to focus handling in a container, so when a child of the container is focused, the adjustment is scrolled to show that widget. This function sets the vertical alignment. See `gtk_scrolled_window_get_vadjustment()` for a typical way of obtaining the adjustment and `gtk_container_set_focus_hadjustment()` for setting the horizontal adjustment.

The adjustments have to be in pixel units and in the same coordinate system as the allocation for immediate children of the container.

Parameters

container	a GtkContainer
adjustment	an adjustment which should be adjusted when the focus is moved among the descendents of `container`

---

gtk_container_get_focus_hadjustment ()

```
GtkAdjustment *
gtk_container_get_focus_hadjustment (GtkContainer *container);
```

Retrieves the horizontal focus adjustment for the container. See `gtk_container_set_focus_hadjustment()`.

Parameters

container	a GtkContainer

Returns

the horizontal focus adjustment, or `NULL` if none has been set.

[nullable][transfer none]

---

gtk_container_set_focus_hadjustment ()

```
void
gtk_container_set_focus_hadjustment (GtkContainer *container,
                                     GtkAdjustment *adjustment);
```

Hooks up an adjustment to focus handling in a container, so when a child of the container is focused, the

adjustment is scrolled to show that widget. This function sets the horizontal alignment. See `gtk_scrolled_window_get_hadjustment()` for a typical way of obtaining the adjustment and `gtk_container_set_focus_vadjustment()` for setting the vertical adjustment.

The adjustments have to be in pixel units and in the same coordinate system as the allocation for immediate children of the container.

Parameters

container	a GtkContainer
adjustment	an adjustment which should be adjusted when the focus is moved among the descendents of `container`

---

gtk_container_resize_children ()

```
void
gtk_container_resize_children (GtkContainer *container);
```

`gtk_container_resize_children` has been deprecated since version 3.10 and should not be used in newly-written code.

Parameters

container	a GtkContainer

---

gtk_container_child_type ()

```
GType
gtk_container_child_type (GtkContainer *container);
```

Returns the type of the children supported by the container.

Note that this may return `G_TYPE_NONE` to indicate that no more children can be added, e.g. for a GtkPaned which already has two children.

Parameters

container	a GtkContainer

Returns

a GType.

---

gtk_container_child_get ()

```
void
gtk_container_child_get (GtkContainer *container,
                         GtkWidget *child,
                         const gchar *first_prop_name,
                         ...);
```

Gets the values of one or more child properties for `child` and `container` .

Parameters

container	a GtkContainer
child	a widget which is a child of `container`
first_prop_name	the name of the first property to get
...	return location for the first property, followed optionally by more name/return location pairs, followed by `NULL`

---

gtk_container_child_set ()

```
void
gtk_container_child_set (GtkContainer *container,
                         GtkWidget *child,
                         const gchar *first_prop_name,
                         ...);
```

Sets one or more child properties for `child` and `container` .

Parameters

container	a GtkContainer
child	a widget which is a child of `container`
first_prop_name	the name of the first property to set
...	a `NULL`-terminated list of property names and values, starting with `first_prop_name`

---

gtk_container_child_get_property ()

```
void
gtk_container_child_get_property (GtkContainer *container,
                                  GtkWidget *child,
                                  const gchar *property_name,
                                  GValue *value);
```

Gets the value of a child property for `child` and `container` .

Parameters

container	a GtkContainer
child	a widget which is a child of `container`
property_name	the name of the property to get
value	a location to return the value

---

gtk_container_child_set_property ()

```
void
gtk_container_child_set_property (GtkContainer *container,
                                  GtkWidget *child,
                                  const gchar *property_name,
                                  const GValue *value);
```

Sets a child property for `child` and `container` .

Parameters

container	a GtkContainer
child	a widget which is a child of `container`
property_name	the name of the property to set
value	the value to set the property to

---

gtk_container_child_get_valist ()

```
void
gtk_container_child_get_valist (GtkContainer *container,
                                GtkWidget *child,
                                const gchar *first_property_name,
                                va_list var_args);
```

Gets the values of one or more child properties for `child` and `container` .

Parameters

container	a GtkContainer
child	a widget which is a child of `container`
first_property_name	the name of the first property to get
var_args	return location for the first property, followed optionally by more name/return location pairs, followed by `NULL`

---

gtk_container_child_set_valist ()

```
void
gtk_container_child_set_valist (GtkContainer *container,
                                GtkWidget *child,
                                const gchar *first_property_name,
                                va_list var_args);
```

Sets one or more child properties for `child` and `container` .

Parameters

container	a GtkContainer
child	a widget which is a child of `container`
first_property_name	the name of the first property to set
var_args	a `NULL`-terminated list of property names and values, starting with `first_prop_name`

---

gtk_container_child_notify ()

```
void
gtk_container_child_notify (GtkContainer *container,
                            GtkWidget *child,
                            const gchar *child_property);
```

Emits a "child-notify" signal for the child property `child_property` on the child.

This is an analogue of `g_object_notify()` for child properties.

Also see `gtk_widget_child_notify()`.

Parameters

container	the GtkContainer
child	the child widget
child_property	the name of a child property installed on the class of `container`

Since: 3.2

---

gtk_container_child_notify_by_pspec ()

```
void
gtk_container_child_notify_by_pspec (GtkContainer *container,
                                     GtkWidget *child,
                                     GParamSpec *pspec);
```

Emits a "child-notify" signal for the child property specified by `pspec` on the child.

This is an analogue of `g_object_notify_by_pspec()` for child properties.

Parameters

container	the GtkContainer
child	the child widget
pspec	the GParamSpec of a child property instealled on the class of `container`

Since: 3.18

---

gtk_container_forall ()

```
void
gtk_container_forall (GtkContainer *container,
                      GtkCallback callback,
                      gpointer callback_data);
```

Invokes `callback` on each direct child of `container` , including children that are considered “internal” (implementation details of the container). “Internal” children generally weren’t added by the user of the container, but were added by the container implementation itself.

Most applications should use `gtk_container_foreach()`, rather than `gtk_container_forall()`.

[virtual forall]

Parameters

container	a GtkContainer	
callback	a callback.	[scope call][closure callback_data]
callback_data	callback user data	

---

gtk_container_get_border_width ()

```
guint
gtk_container_get_border_width (GtkContainer *container);
```

Retrieves the border width of the container. See `gtk_container_set_border_width()`.

Parameters

container	a GtkContainer

Returns

the current border width

---

gtk_container_set_border_width ()

```
void
gtk_container_set_border_width (GtkContainer *container,
```

```
                                guint border_width);
```

Sets the border width of the container.

The border width of a container is the amount of space to leave around the outside of the container. The only exception to this is GtkWindow; because toplevel windows can't leave space outside, they leave the space inside. The border is added on all sides of the container. To add space to only one side, use a specific "margin" property on the child widget, for example "margin-top".

Parameters

container	a GtkContainer
border_width	amount of blank space to leave outside the container. Valid values are in the range 0-65535 pixels.

---

gtk_container_propagate_draw ()

```
void
gtk_container_propagate_draw (GtkContainer *container,
                              GtkWidget *child,
                              cairo_t *cr);
```

When a container receives a call to the draw function, it must send synthetic "draw" calls to all children that don't have their own GdkWindows. This function provides a convenient way of doing this. A container, when it receives a call to its "draw" function, calls `gtk_container_propagate_draw()` once for each child, passing in the `cr` the container received.

gtk_container_propagate_draw() takes care of translating the origin of `cr` , and deciding whether the draw needs to be sent to the child. It is a convenient and optimized way of getting the same effect as calling `gtk_widget_draw()` on the child directly.

In most cases, a container can simply either inherit the "draw" implementation from GtkContainer, or do some drawing and then chain to the ::draw implementation from GtkContainer.

Parameters

container	a GtkContainer
child	a child of `container`
cr	Cairo context as passed to the container. If you want to use `cr` in container's draw function, consider using `cairo_save()` and `cairo_restore()` before calling this function.

---

gtk_container_get_focus_chain ()

```
gboolean
gtk_container_get_focus_chain (GtkContainer *container,
```

```
                                GList **focusable_widgets);
```

`gtk_container_get_focus_chain` has been deprecated since version 3.24 and should not be used in newly-written code.

For overriding focus behavior, use the GtkWidgetClass::focus signal.

Retrieves the focus chain of the container, if one has been set explicitly. If no focus chain has been explicitly set, GTK+ computes the focus chain based on the positions of the children. In that case, GTK+ stores `NULL` in `focusable_widgets` and returns `FALSE`.

Parameters

container	a GtkContainer	
focusable_widgets	location to store the focus chain of the container, or `NULL`. You should free this list using `g_list_free()` when you are done with it, however no additional reference count is added to the individual widgets in the focus chain.	[element-type GtkWidget][out][transfer container]

Returns

`TRUE` if the focus chain of the container has been set explicitly.

---

gtk_container_set_focus_chain ()

```
void
gtk_container_set_focus_chain (GtkContainer *container,
                               GList *focusable_widgets);
```

`gtk_container_set_focus_chain` has been deprecated since version 3.24 and should not be used in newly-written code.

For overriding focus behavior, use the GtkWidgetClass::focus signal.

Sets a focus chain, overriding the one computed automatically by GTK+.

In principle each widget in the chain should be a descendant of the container, but this is not enforced by this method, since it’s allowed to set the focus chain before you pack the widgets, or have a widget in the chain that isn’t always packed. The necessary checks are done when the focus chain is actually traversed.

Parameters

container	a GtkContainer	
focusable_widgets	the new focus chain.	[transfer none][element-type GtkWidget]

---

gtk_container_unset_focus_chain ()

```
void
gtk_container_unset_focus_chain (GtkContainer *container);
```

`gtk_container_unset_focus_chain` has been deprecated since version 3.24 and should not be used in newly-written code.

For overriding focus behavior, use the GtkWidgetClass::focus signal.

Removes a focus chain explicitly set with `gtk_container_set_focus_chain()`.

Parameters

container	a GtkContainer

---

gtk_container_class_find_child_property ()

```
GParamSpec *
gtk_container_class_find_child_property
                                (GObjectClass *cclass,
                                 const gchar *property_name);
```

Finds a child property of a container class by name.

Parameters

cclass	a GtkContainerClass.	[type GtkContainerClass]
property_name	the name of the child property to find	

Returns

the GParamSpec of the child property or `NULL` if `class` has no child property with that name.

[nullable][transfer none]

---

gtk_container_class_install_child_property ()

```
void
gtk_container_class_install_child_property
                                (GtkContainerClass *cclass,
                                 guint property_id,
                                 GParamSpec *pspec);
```

Installs a child property on a container class.

Parameters

cclass	a GtkContainerClass
property_id	the id for the property

pspec	the GParamSpec for the property

gtk_container_class_install_child_properties ()

```
void
gtk_container_class_install_child_properties
                                (GtkContainerClass *cclass,
                                 guint n_pspecs,
                                 GParamSpec **pspecs);
```

Installs child properties on a container class.

Parameters

cclass	a GtkContainerClass	
n_pspecs	the length of the GParamSpec array	
pspecs	the GParamSpec array defining the new child properties.	[array length=n_pspecs]

Since: 3.18

gtk_container_class_list_child_properties ()

```
GParamSpec **
gtk_container_class_list_child_properties
                                (GObjectClass *cclass,
                                 guint *n_properties);
```

Returns all child properties of a container class.

Parameters

cclass	a GtkContainerClass.	[type GtkContainerClass]
n_properties	location to return the number of child properties found	

Returns

a newly allocated `NULL`-terminated array of GParamSpec*. The array must be freed with `g_free()`.

[array length=n_properties][transfer container]

gtk_container_class_handle_border_width ()

```
void
gtk_container_class_handle_border_width
                                (GtkContainerClass *klass);
```

Modifies a subclass of GtkContainerClass to automatically add and remove the border-width setting on GtkContainer. This allows the subclass to ignore the border width in its size request and allocate methods. The

intent is for a subclass to invoke this in its class_init function.

gtk_container_class_handle_border_width() is necessary because it would break API too badly to make this behavior the default. So subclasses must “opt in” to the parent class handling border_width for them.

Parameters

klass	the class struct of a GtkContainer subclass

Types and Values

struct GtkContainer

```
struct GtkContainer;
```

---

struct GtkContainerClass

```
struct GtkContainerClass {
  GtkWidgetClass parent_class;


  void    (*add)              (GtkContainer      *container,
                               GtkWidget   *widget);
  void    (*remove)           (GtkContainer      *container,
                               GtkWidget   *widget);
  void    (*check_resize) (GtkContainer    *container);
  void    (*forall)           (GtkContainer      *container,
                               gboolean     include_internals,
                               GtkCallback        callback,
                               gpointer     callback_data);
  void    (*set_focus_child) (GtkContainer      *container,
                               GtkWidget   *child);
  GType   (*child_type)       (GtkContainer      *container);
  gchar*  (*composite_name) (GtkContainer  *container,
                               GtkWidget   *child);
  void    (*set_child_property) (GtkContainer    *container,
                               GtkWidget       *child,
                               guint            property_id,
                               const GValue    *value,
                               GParamSpec      *pspec);
  void    (*get_child_property) (GtkContainer    *container,
                                 GtkWidget       *child,
                               guint            property_id,
                               GValue          *value,
                               GParamSpec      *pspec);
  GtkWidgetPath * (*get_path_for_child) (GtkContainer *container,
                                         GtkWidget    *child);
};
```

Base class for containers.

Members

`add` ()	Signal emitted when a widget is added to container.
`remove` ()	Signal emitted when a widget is removed from container.
`check_resize` ()	Signal emitted when a size recalculation is needed.
`forall` ()	Invokes callback on each child of container. The callback handler may remove the child.
`set_focus_child` ()	Sets the focused child of container.
`child_type` ()	Returns the type of the children supported by the container.
`composite_name` ()	Gets a widget's composite name. Deprecated: 3.10.
`set_child_property` ()	Set a property on a child of container.
`get_child_property` ()	Get a property from a child of container.
`get_path_for_child` ()	Get path representing entire widget hierarchy from the toplevel down to and including `child` .

---

enum GtkResizeMode

Members

GTK_RESIZE_PARENT	Pass resize request to the parent
GTK_RESIZE_QUEUE	Queue resizes on this widget
GTK_RESIZE_IMMEDIATE	Resize immediately. Deprecated.

Property Details

The `"border-width"` property

`"border-width"` `guint`

The width of the empty border outside the containers children.

Flags: Read / Write

Allowed values: <= 65535

Default value: 0

The “child” property

```
  “child”                    GtkWidget *
```

Can be used to add a new child to the container.

Flags: Write

The “resize-mode” property

```
  “resize-mode”              GtkResizeMode
```

Specify how resize events are handled.

Flags: Read / Write

Default value: GTK_RESIZE_PARENT

Signal Details

The “add” signal

```
void
user_function (GtkContainer *container,
               GtkWidget    *widget,
               gpointer      user_data)
```

Flags: Run First

The “check-resize” signal

```
void
user_function (GtkContainer *container,
               gpointer      user_data)
```

Flags: Run Last

The “remove” signal

```
void
user_function (GtkContainer *container,
               GtkWidget    *widget,
               gpointer      user_data)
```

Flags: Run First

The “set-focus-child” signal

```
void
user_function (GtkContainer *container,
```

```
                GtkWidget    *widget,
                gpointer      user_data)
```

Flags: Run First

GtkBin

GtkBin — A container with just one child

Functions

GtkWidget *	gtk_bin_get_child ()

Types and Values

struct	GtkBin
struct	GtkBinClass

Object Hierarchy

```
    GObject
    ╰── GInitiallyUnowned
        ╰── GtkWidget
            ╰── GtkContainer
                ╰── GtkBin
                    ├── GtkWindow
                    ├── GtkActionBar
                    ├── GtkAlignment
                    ├── GtkComboBox
                    ├── GtkFrame
                    ├── GtkButton
                    ├── GtkMenuItem
                    ├── GtkEventBox
                    ├── GtkExpander
                    ├── GtkFlowBoxChild
                    ├── GtkHandleBox
                    ├── GtkListBoxRow
                    ├── GtkToolItem
                    ├── GtkOverlay
                    ├── GtkScrolledWindow
                    ├── GtkPopover
                    ├── GtkRevealer
                    ├── GtkSearchBar
                    ├── GtkStackSidebar
                    ╰── GtkViewport
```

Implemented Interfaces

GtkBin implements AtkImplementorIface and GtkBuildable.

Includes

```
#include <gtk/gtk.h>
```

Description

The GtkBin widget is a container with just one child. It is not very useful itself, but it is useful for deriving subclasses, since it provides common code needed for handling a single child widget.

Many GTK+ widgets are subclasses of GtkBin, including GtkWindow, GtkButton, GtkFrame, GtkHandleBox or GtkScrolledWindow.

Functions

gtk_bin_get_child ()

```
GtkWidget *
gtk_bin_get_child (GtkBin *bin);
```

Gets the child of the GtkBin, or `NULL` if the bin contains no child widget. The returned widget does not have a reference added, so you do not need to unref it.

Parameters

bin	a GtkBin

Returns

the child of `bin` , or `NULL` if it does not have a child.

[transfer none][nullable]

Types and Values

struct GtkBin

```
struct GtkBin;
```

---

struct GtkBinClass

```
struct GtkBinClass {
  GtkContainerClass parent_class;
};
```

Members

GtkMenuShell

GtkMenuShell — A base class for menu objects

Functions

void	gtk_menu_shell_append ()
void	gtk_menu_shell_prepend ()
void	gtk_menu_shell_insert ()
void	gtk_menu_shell_deactivate ()
void	gtk_menu_shell_select_item ()
void	gtk_menu_shell_select_first ()
void	gtk_menu_shell_deselect ()
void	gtk_menu_shell_activate_item ()
void	gtk_menu_shell_cancel ()
void	gtk_menu_shell_set_take_focus ()
gboolean	gtk_menu_shell_get_take_focus ()
GtkWidget *	gtk_menu_shell_get_selected_item ()
GtkWidget *	gtk_menu_shell_get_parent_shell ()
void	gtk_menu_shell_bind_model ()

Properties

gboolean	take-focus	Read / Write

Signals

void	activate-current	Action
void	cancel	Action
void	cycle-focus	Action
void	deactivate	Run First
void	insert	Run First
void	move-current	Action
gboolean	move-selected	Run Last
void	selection-done	Run First

Types and Values

struct	GtkMenuShell
enum	GtkMenuDirectionType

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkContainer
            └── GtkMenuShell
                ├── GtkMenuBar
                └── GtkMenu
```

Implemented Interfaces

GtkMenuShell implements AtkImplementorIface and GtkBuildable.

Includes

```
#include <gtk/gtk.h>
```

Description

A GtkMenuShell is the abstract base class used to derive the GtkMenu and GtkMenuBar subclasses.

A GtkMenuShell is a container of GtkMenuItem objects arranged in a list which can be navigated, selected, and activated by the user to perform application functions. A GtkMenuItem can have a submenu associated with it, allowing for nested hierarchical menus.

Terminology

A menu item can be “selected”, this means that it is displayed in the prelight state, and if it has a submenu, that submenu will be popped up.

A menu is “active” when it is visible onscreen and the user is selecting from it. A menubar is not active until the user clicks on one of its menuitems. When a menu is active, passing the mouse over a submenu will pop it up.

There is also is a concept of the current menu and a current menu item. The current menu item is the selected menu item that is furthest down in the hierarchy. (Every active menu shell does not necessarily contain a selected menu item, but if it does, then the parent menu shell must also contain a selected menu item.) The current menu is the menu that contains the current menu item. It will always have a GTK grab and receive all key presses.

Functions

gtk_menu_shell_append ()

```
void
gtk_menu_shell_append (GtkMenuShell *menu_shell,
                       GtkWidget *child);
```

Adds a new GtkMenuItem to the end of the menu shell's item list.

Parameters

menu_shell	a GtkMenuShell	
child	The GtkMenuItem to add.	[type Gtk.MenuItem]

gtk_menu_shell_prepend ()

```
void
gtk_menu_shell_prepend (GtkMenuShell *menu_shell,
                        GtkWidget *child);
```

Adds a new GtkMenuItem to the beginning of the menu shell's item list.

Parameters

menu_shell	a GtkMenuShell
child	The GtkMenuItem to add

---

gtk_menu_shell_insert ()

```
void
gtk_menu_shell_insert (GtkMenuShell *menu_shell,
                       GtkWidget *child,
                       gint position);
```

Adds a new GtkMenuItem to the menu shell’s item list at the position indicated by `position` .

Parameters

menu_shell	a GtkMenuShell
child	The GtkMenuItem to add
position	The position in the item list where `child` is added. Positions are numbered from 0 to n-1

---

gtk_menu_shell_deactivate ()

```
void
gtk_menu_shell_deactivate (GtkMenuShell *menu_shell);
```

Deactivates the menu shell.

Typically this results in the menu shell being erased from the screen.

Parameters

menu_shell	a GtkMenuShell

---

gtk_menu_shell_select_item ()

```
void
gtk_menu_shell_select_item (GtkMenuShell *menu_shell,
                            GtkWidget *menu_item);
```

Selects the menu item from the menu shell.

Parameters

menu_shell	a GtkMenuShell
menu_item	The GtkMenuItem to select

---

gtk_menu_shell_select_first ()

```
void
gtk_menu_shell_select_first (GtkMenuShell *menu_shell,
                             gboolean search_sensitive);
```

Select the first visible or selectable child of the menu shell; don’t select tearoff items unless the only item is a tearoff item.

Parameters

menu_shell	a GtkMenuShell
search_sensitive	if TRUE, search for the first selectable menu item, otherwise select nothing if the first item isn’t sensitive. This should be FALSE if the menu is being popped up initially.

Since: 2.2

---

gtk_menu_shell_deselect ()

```
void
gtk_menu_shell_deselect (GtkMenuShell *menu_shell);
```

Deselects the currently selected item from the menu shell, if any.

Parameters

menu_shell	a GtkMenuShell

---

gtk_menu_shell_activate_item ()

```
void
gtk_menu_shell_activate_item (GtkMenuShell *menu_shell,
                              GtkWidget *menu_item,
                              gboolean force_deactivate);
```

Activates the menu item within the menu shell.

Parameters

menu_shell	a GtkMenuShell
menu_item	the GtkMenuItem to activate
force_deactivate	if TRUE, force the deactivation of the menu shell after the menu item is activated

---

gtk_menu_shell_cancel ()

```
void
gtk_menu_shell_cancel (GtkMenuShell *menu_shell);
```

Cancels the selection within the menu shell.

Parameters

menu_shell	a GtkMenuShell

Since: 2.4

---

gtk_menu_shell_set_take_focus ()

```
void
gtk_menu_shell_set_take_focus (GtkMenuShell *menu_shell,
                               gboolean take_focus);
```

If take_focus is TRUE (the default) the menu shell will take the keyboard focus so that it will receive all keyboard events which is needed to enable keyboard navigation in menus.

Setting take_focus to FALSE is useful only for special applications like virtual keyboard implementations which should not take keyboard focus.

The take_focus state of a menu or menu bar is automatically propagated to submenus whenever a submenu is popped up, so you don’t have to worry about recursively setting it for your entire menu hierarchy. Only when programmatically picking a submenu and popping it up manually, the take_focus property of the submenu needs to be set explicitly.

Note that setting it to FALSE has side-effects:

If the focus is in some other app, it keeps the focus and keynav in the menu doesn’t work. Consequently, keynav on the menu will only work if the focus is on some toplevel owned by the onscreen keyboard.

To avoid confusing the user, menus with take_focus set to FALSE should not display mnemonics or accelerators, since it cannot be guaranteed that they will work.

See also gdk_keyboard_grab()

Parameters

menu_shell	a GtkMenuShell
take_focus	TRUE if the menu shell should take

the keyboard focus on popup

Since: 2.8

---

gtk_menu_shell_get_take_focus ()

```
gboolean
gtk_menu_shell_get_take_focus (GtkMenuShell *menu_shell);
```

Returns `TRUE` if the menu shell will take the keyboard focus on popup.

Parameters

menu_shell	a GtkMenuShell

Returns

`TRUE` if the menu shell will take the keyboard focus on popup.

Since: 2.8

---

gtk_menu_shell_get_selected_item ()

```
GtkWidget *
gtk_menu_shell_get_selected_item (GtkMenuShell *menu_shell);
```

Gets the currently selected item.

Parameters

menu_shell	a GtkMenuShell

Returns

the currently selected item.

[transfer none]

Since: 3.0

---

gtk_menu_shell_get_parent_shell ()

```
GtkWidget *
gtk_menu_shell_get_parent_shell (GtkMenuShell *menu_shell);
```

Gets the parent menu shell.

The parent menu shell of a submenu is the GtkMenu or GtkMenuBar from which it was opened up.

Parameters

menu_shell	a GtkMenuShell

Returns

the parent GtkMenuShell.

[transfer none]

Since: 3.0

---

gtk_menu_shell_bind_model ()

```
void
gtk_menu_shell_bind_model (GtkMenuShell *menu_shell,
                           GMenuModel *model,
                           const gchar *action_namespace,
                           gboolean with_separators);
```

Establishes a binding between a GtkMenuShell and a GMenuModel.

The contents of `shell` are removed and then refilled with menu items according to `model` . When `model` changes, `shell` is updated. Calling this function twice on `shell` with different `model` will cause the first binding to be replaced with a binding to the new model. If `model` is `NULL` then any previous binding is undone and all children are removed.

`with_separators` determines if toplevel items (eg: sections) have separators inserted between them. This is typically desired for menus but doesn't make sense for menubars.

If `action_namespace` is non-`NULL` then the effect is as if all actions mentioned in the `model` have their names prefixed with the namespace, plus a dot. For example, if the action "quit" is mentioned and `action_namespace` is "app" then the effective action name is "app.quit".

This function uses GtkActionable to define the action name and target values on the created menu items. If you want to use an action group other than "app" and "win", or if you want to use a GtkMenuShell outside of a GtkApplicationWindow, then you will need to attach your own action group to the widget hierarchy using `gtk_widget_insert_action_group()`. As an example, if you created a group with a "quit" action and inserted it with the name "mygroup" then you would use the action name "mygroup.quit" in your GMenuModel.

For most cases you are probably better off using `gtk_menu_new_from_model()` or `gtk_menu_bar_new_from_model()` or just directly passing the GMenuModel to `gtk_application_set_app_menu()` or `gtk_application_set_menubar()`.

Parameters

menu_shell	a GtkMenuShell
model	the GMenuModel to bind to or `NULL` [allow-none] to remove binding.
action_namespace	the namespace for actions in `model` . [allow-none]
with_separators	`TRUE` if toplevel items in `shell` should have separators between them

Since: 3.6

Types and Values

struct GtkMenuShell

```
struct GtkMenuShell;
```

---

enum GtkMenuDirectionType

An enumeration representing directional movements within a menu.

Members

GTK_MENU_DIR_PARENT	To the parent menu shell
GTK_MENU_DIR_CHILD	To the submenu, if any, associated with the item
GTK_MENU_DIR_NEXT	To the next menu item
GTK_MENU_DIR_PREV	To the previous menu item

Property Details

The `"take-focus"` property

```
  "take-focus"               gboolean
```

A boolean that determines whether the menu and its submenus grab the keyboard focus. See `gtk_menu_shell_set_take_focus()` and `gtk_menu_shell_get_take_focus()`.

Flags: Read / Write

Default value: TRUE

Since: 2.8

Signal Details

The `"activate-current"` signal

```
void
user_function (GtkMenuShell *menushell,
               gboolean      force_hide,
               gpointer      user_data)
```

An action signal that activates the current menu item within the menu shell.

Parameters

menushell	the object which received the signal
force_hide	if `TRUE`, hide the menu after activating the menu item
user_data	user data set when the signal handler was connected.

Flags: Action

The `"cancel"` signal

```
void
user_function (GtkMenuShell *menushell,
               gpointer      user_data)
```

An action signal which cancels the selection within the menu shell. Causes the "selection-done" signal to be emitted.

Parameters

menushell	the object which received the signal
user_data	user data set when the signal handler was connected.

Flags: Action

The `"cycle-focus"` signal

```
void
user_function (GtkMenuShell    *menushell,
               GtkDirectionType direction,
               gpointer         user_data)
```

A keybinding signal which moves the focus in the given `direction` .

Parameters

menushell	the object which received the signal
direction	the direction to cycle in
user_data	user data set when the signal handler was connected.

Flags: Action

The `"deactivate"` signal

```
void
user_function (GtkMenuShell *menushell,
```

```
                gpointer      user_data)
```
This signal is emitted when a menu shell is deactivated.

Parameters

menushell	the object which received the signal
user_data	user data set when the signal handler was connected.

Flags: Run First

The `"insert"` signal

```
void
user_function (GtkMenuShell *menu_shell,
               GtkWidget    *child,
               gint          position,
               gpointer      user_data)
```
The ::insert signal is emitted when a new GtkMenuItem is added to a GtkMenuShell. A separate signal is used instead of GtkContainer::add because of the need for an additional position parameter.

The inverse of this signal is the GtkContainer::removed signal.

Parameters

menu_shell	the object on which the signal is emitted
child	the GtkMenuItem that is being inserted
position	the position at which the insert occurs
user_data	user data set when the signal handler was connected.

Flags: Run First

Since: 3.2

The `"move-current"` signal

```
void
user_function (GtkMenuShell        *menushell,
               GtkMenuDirectionType direction,
               gpointer             user_data)
```
An keybinding signal which moves the current menu item in the direction specified by `direction` .

Parameters

menushell	the object which received the signal

direction	the direction to move
user_data	user data set when the signal handler was connected.

Flags: Action

The “move-selected” signal

```
gboolean
user_function (GtkMenuShell *menu_shell,
               gint          distance,
               gpointer      user_data)
```

The ::move-selected signal is emitted to move the selection to another item.

Parameters

menu_shell	the object on which the signal is emitted
distance	+1 to move to the next item, -1 to move to the previous
user_data	user data set when the signal handler was connected.

Returns

`TRUE` to stop the signal emission, `FALSE` to continue

Flags: Run Last

Since: 2.12

The “selection-done” signal

```
void
user_function (GtkMenuShell *menushell,
               gpointer      user_data)
```

This signal is emitted when a selection has been completed within a menu shell.

Parameters

menushell	the object which received the signal
user_data	user data set when the signal handler was connected.

Flags: Run First

GtkRange

GtkRange — Base class for widgets which visualize an adjustment

Functions

gdouble	gtk_range_get_fill_level ()
gboolean	gtk_range_get_restrict_to_fill_level ()
gboolean	gtk_range_get_show_fill_level ()
void	gtk_range_set_fill_level ()
void	gtk_range_set_restrict_to_fill_level ()
void	gtk_range_set_show_fill_level ()
GtkAdjustment *	gtk_range_get_adjustment ()
void	gtk_range_set_adjustment ()
gboolean	gtk_range_get_inverted ()
void	gtk_range_set_inverted ()
gdouble	gtk_range_get_value ()
void	gtk_range_set_value ()
void	gtk_range_set_increments ()
void	gtk_range_set_range ()
gint	gtk_range_get_round_digits ()
void	gtk_range_set_round_digits ()
void	gtk_range_set_lower_stepper_sensitivity ()
GtkSensitivityType	gtk_range_get_lower_stepper_sensitivity ()
void	gtk_range_set_upper_stepper_sensitivity ()
GtkSensitivityType	gtk_range_get_upper_stepper_sensitivity ()
gboolean	gtk_range_get_flippable ()
void	gtk_range_set_flippable ()
gint	gtk_range_get_min_slider_size ()
void	gtk_range_get_range_rect ()
void	gtk_range_get_slider_range ()
gboolean	gtk_range_get_slider_size_fixed ()
void	gtk_range_set_min_slider_size ()
void	gtk_range_set_slider_size_fixed ()

Properties

GtkAdjustment *	adjustment	Read / Write / Construct
gdouble	fill-level	Read / Write
gboolean	inverted	Read / Write
GtkSensitivityType	lower-stepper-sensitivity	Read / Write
gboolean	restrict-to-fill-level	Read / Write
gint	round-digits	Read / Write
gboolean	show-fill-level	Read / Write
GtkSensitivityType	upper-stepper-sensitivity	Read / Write

Style Properties

gint	arrow-displacement-x	Read

gint	arrow-displacement-y	Read
gfloat	arrow-scaling	Read
gint	slider-width	Read
gint	stepper-size	Read
gint	stepper-spacing	Read
gint	trough-border	Read
gboolean	trough-under-steppers	Read

Signals

void	adjust-bounds	Run Last
gboolean	change-value	Run Last
void	move-slider	Action
void	value-changed	Run Last

Types and Values

struct	GtkRange
enum	GtkSensitivityType

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkRange
            ├── GtkScale
            └── GtkScrollbar
```

Implemented Interfaces

GtkRange implements AtkImplementorIface, GtkBuildable and GtkOrientable.

Includes

```
#include <gtk/gtk.h>
```

Description

GtkRange is the common base class for widgets which visualize an adjustment, e.g GtkScale or GtkScrollbar.

Apart from signals for monitoring the parameters of the adjustment, GtkRange provides properties and methods for influencing the sensitivity of the “steppers”. It also provides properties and methods for setting a “fill level” on range widgets. See `gtk_range_set_fill_level()`.

Functions

gtk_range_get_fill_level ()

```
gdouble
gtk_range_get_fill_level (GtkRange *range);
```

Gets the current position of the fill level indicator.

Parameters

range	A GtkRange

Returns

The current fill level

Since: 2.12

---

gtk_range_get_restrict_to_fill_level ()

```
gboolean
gtk_range_get_restrict_to_fill_level (GtkRange *range);
```

Gets whether the range is restricted to the fill level.

Parameters

range	A GtkRange

Returns

`TRUE` if `range` is restricted to the fill level.

Since: 2.12

---

gtk_range_get_show_fill_level ()

```
gboolean
gtk_range_get_show_fill_level (GtkRange *range);
```

Gets whether the range displays the fill level graphically.

Parameters

range	A GtkRange

Returns

`TRUE` if `range` shows the fill level.

Since: 2.12

gtk_range_set_fill_level ()

```
void
gtk_range_set_fill_level (GtkRange *range,
                          gdouble fill_level);
```

Set the new position of the fill level indicator.

The “fill level” is probably best described by its most prominent use case, which is an indicator for the amount of pre-buffering in a streaming media player. In that use case, the value of the range would indicate the current play position, and the fill level would be the position up to which the file/stream has been downloaded.

This amount of prebuffering can be displayed on the range’s trough and is themeable separately from the trough. To enable fill level display, use `gtk_range_set_show_fill_level()`. The range defaults to not showing the fill level.

Additionally, it’s possible to restrict the range’s slider position to values which are smaller than the fill level. This is controller by `gtk_range_set_restrict_to_fill_level()` and is by default enabled.

Parameters

range	a GtkRange
fill_level	the new position of the fill level indicator

Since: 2.12

---

gtk_range_set_restrict_to_fill_level ()

```
void
gtk_range_set_restrict_to_fill_level (GtkRange *range,
                                      gboolean restrict_to_fill_level);
```

Sets whether the slider is restricted to the fill level. See `gtk_range_set_fill_level()` for a general description of the fill level concept.

Parameters

range	A GtkRange
restrict_to_fill_level	Whether the fill level restricts slider movement.

Since: 2.12

---

gtk_range_set_show_fill_level ()

```
void
gtk_range_set_show_fill_level (GtkRange *range,
                               gboolean show_fill_level);
```

Sets whether a graphical fill level is show on the trough. See `gtk_range_set_fill_level()` for a general description of the fill level concept.

Parameters

range	A GtkRange
show_fill_level	Whether a fill level indicator graphics is shown.

Since: 2.12

---

gtk_range_get_adjustment ()

```
GtkAdjustment *
gtk_range_get_adjustment (GtkRange *range);
```

Get the GtkAdjustment which is the “model” object for GtkRange. See `gtk_range_set_adjustment()` for details. The return value does not have a reference added, so should not be unreferenced.

Parameters

range	a GtkRange

Returns

a GtkAdjustment.

[transfer none]

---

gtk_range_set_adjustment ()

```
void
gtk_range_set_adjustment (GtkRange *range,
                          GtkAdjustment *adjustment);
```

Sets the adjustment to be used as the “model” object for this range widget. The adjustment indicates the current range value, the minimum and maximum range values, the step/page increments used for keybindings and scrolling, and the page size. The page size is normally 0 for GtkScale and nonzero for GtkScrollbar, and indicates the size of the visible area of the widget being scrolled. The page size affects the size of the scrollbar slider.

Parameters

range	a GtkRange
adjustment	a GtkAdjustment

---

gtk_range_get_inverted ()

```
gboolean
gtk_range_get_inverted (GtkRange *range);
```

Gets the value set by `gtk_range_set_inverted()`.

Parameters

range	a GtkRange

Returns

`TRUE` if the range is inverted

---

gtk_range_set_inverted ()

```
void
gtk_range_set_inverted (GtkRange *range,
                        gboolean setting);
```

Ranges normally move from lower to higher values as the slider moves from top to bottom or left to right. Inverted ranges have higher values at the top or on the right rather than on the bottom or left.

Parameters

range	a GtkRange
setting	`TRUE` to invert the range

---

gtk_range_get_value ()

```
gdouble
gtk_range_get_value (GtkRange *range);
```

Gets the current value of the range.

Parameters

range	a GtkRange

Returns

current value of the range.

---

gtk_range_set_value ()

```
void
```

```
gtk_range_set_value (GtkRange *range,
                     gdouble value);
```

Sets the current value of the range; if the value is outside the minimum or maximum range values, it will be clamped to fit inside them. The range emits the "value-changed" signal if the value changes.

Parameters

range	a GtkRange
value	new value of the range

---

gtk_range_set_increments ()

```
void
gtk_range_set_increments (GtkRange *range,
                          gdouble step,
                          gdouble page);
```

Sets the step and page sizes for the range. The step size is used when the user clicks the GtkScrollbar arrows or moves GtkScale via arrow keys. The page size is used for example when moving via Page Up or Page Down keys.

Parameters

range	a GtkRange
step	step size
page	page size

---

gtk_range_set_range ()

```
void
gtk_range_set_range (GtkRange *range,
                     gdouble min,
                     gdouble max);
```

Sets the allowable values in the GtkRange, and clamps the range value to be between `min` and `max` . (If the range has a non-zero page size, it is clamped between `min` and `max` - page-size.)

Parameters

range	a GtkRange
min	minimum range value
max	maximum range value

---

gtk_range_get_round_digits ()

```
gint
gtk_range_get_round_digits (GtkRange *range);
```

Gets the number of digits to round the value to when it changes. See “change-value”.

Parameters

range	a GtkRange

Returns

the number of digits to round to

Since: 2.24

---

gtk_range_set_round_digits ()

```
void
gtk_range_set_round_digits (GtkRange *range,
                            gint round_digits);
```

Sets the number of digits to round the value to when it changes. See “change-value”.

Parameters

range	a GtkRange
round_digits	the precision in digits, or -1

Since: 2.24

---

gtk_range_set_lower_stepper_sensitivity ()

```
void
gtk_range_set_lower_stepper_sensitivity
                                (GtkRange *range,
                                 GtkSensitivityType sensitivity);
```

Sets the sensitivity policy for the stepper that points to the 'lower' end of the GtkRange’s adjustment.

Parameters

range	a GtkRange
sensitivity	the lower stepper’s sensitivity policy.

Since: 2.10

---

gtk_range_get_lower_stepper_sensitivity ()

```
GtkSensitivityType
gtk_range_get_lower_stepper_sensitivity
                                (GtkRange *range);
```

Gets the sensitivity policy for the stepper that points to the 'lower' end of the GtkRange’s adjustment.

Parameters

range	a GtkRange

Returns

The lower stepper’s sensitivity policy.

Since: 2.10

---

gtk_range_set_upper_stepper_sensitivity ()

```
void
gtk_range_set_upper_stepper_sensitivity
                                (GtkRange *range,
                                 GtkSensitivityType sensitivity);
```

Sets the sensitivity policy for the stepper that points to the 'upper' end of the GtkRange’s adjustment.

Parameters

range	a GtkRange
sensitivity	the upper stepper’s sensitivity policy.

Since: 2.10

---

gtk_range_get_upper_stepper_sensitivity ()

```
GtkSensitivityType
gtk_range_get_upper_stepper_sensitivity
                                (GtkRange *range);
```

Gets the sensitivity policy for the stepper that points to the 'upper' end of the GtkRange’s adjustment.

Parameters

range	a GtkRange

Returns

The upper stepper’s sensitivity policy.

Since: 2.10

---

gtk_range_get_flippable ()

```
gboolean
gtk_range_get_flippable (GtkRange *range);
```

Gets the value set by `gtk_range_set_flippable()`.

Parameters

range	a GtkRange

Returns

`TRUE` if the range is flippable

Since: 2.18

---

gtk_range_set_flippable ()

```
void
gtk_range_set_flippable (GtkRange *range,
                         gboolean flippable);
```

If a range is flippable, it will switch its direction if it is horizontal and its direction is `GTK_TEXT_DIR_RTL`.

See `gtk_widget_get_direction()`.

Parameters

range	a GtkRange
flippable	`TRUE` to make the range flippable

Since: 2.18

---

gtk_range_get_min_slider_size ()

```
gint
gtk_range_get_min_slider_size (GtkRange *range);
```

`gtk_range_get_min_slider_size` has been deprecated since version 3.20 and should not be used in newly-written code.

Use the min-height/min-width CSS properties on the slider node.

This function is useful mainly for GtkRange subclasses.

See `gtk_range_set_min_slider_size()`.

Parameters

range	a GtkRange

Returns

The minimum size of the range’s slider.

Since: 2.20

---

gtk_range_get_range_rect ()

```
void
gtk_range_get_range_rect (GtkRange *range,
                          GdkRectangle *range_rect);
```

This function returns the area that contains the range’s trough and its steppers, in widget->window coordinates.

This function is useful mainly for GtkRange subclasses.

Parameters

range	a GtkRange	
range_rect	return location for the range rectangle.	[out]

Since: 2.20

---

gtk_range_get_slider_range ()

```
void
gtk_range_get_slider_range (GtkRange *range,
                            gint *slider_start,
                            gint *slider_end);
```

This function returns sliders range along the long dimension, in widget->window coordinates.

This function is useful mainly for GtkRange subclasses.

Parameters

range	a GtkRange	
slider_start	return location for the slider's start, or `NULL`.	[out][allow-none]
slider_end	return location for the slider's end, or `NULL`.	[out][allow-none]

Since: 2.20

---

gtk_range_get_slider_size_fixed ()

```
gboolean
gtk_range_get_slider_size_fixed (GtkRange *range);
```

This function is useful mainly for GtkRange subclasses.

See `gtk_range_set_slider_size_fixed()`.

Parameters

range	a GtkRange

Returns

whether the range's slider has a fixed size.

Since: 2.20

---

gtk_range_set_min_slider_size ()

```
void
gtk_range_set_min_slider_size (GtkRange *range,
                               gint min_size);
```

`gtk_range_set_min_slider_size` has been deprecated since version 3.20 and should not be used in newly-written code.

Use the min-height/min-width CSS properties on the slider node.

Sets the minimum size of the range's slider.

This function is useful mainly for GtkRange subclasses.

Parameters

range	a GtkRange
min_size	The slider's minimum size

Since: 2.20

---

gtk_range_set_slider_size_fixed ()

```
void
gtk_range_set_slider_size_fixed (GtkRange *range,
                                 gboolean size_fixed);
```

Sets whether the range's slider has a fixed size, or a size that depends on its adjustment's page size.

This function is useful mainly for GtkRange subclasses.

Parameters

range	a GtkRange
size_fixed	`TRUE` to make the slider size constant

Since: 2.20

Types and Values

struct GtkRange

```
struct GtkRange;
```

---

enum GtkSensitivityType

Determines how GTK+ handles the sensitivity of stepper arrows at the end of range widgets.

Members

GTK_SENSITIVITY_AUTO	The arrow is made insensitive if the thumb is at the end
GTK_SENSITIVITY_ON	The arrow is always sensitive
GTK_SENSITIVITY_OFF	The arrow is always insensitive

Property Details

The “adjustment” property

```
  “adjustment”               GtkAdjustment *
```

The GtkAdjustment that contains the current value of this range object.

Flags: Read / Write / Construct

The “fill-level” property

```
  “fill-level”               gdouble
```

The fill level (e.g. prebuffering of a network stream). See `gtk_range_set_fill_level()`.

Flags: Read / Write

Default value: 1.79769e+308

Since: 2.12

The “inverted” property

```
  “inverted”                 gboolean
```

Invert direction slider moves to increase range value.

Flags: Read / Write

Default value: FALSE

The “lower-stepper-sensitivity” property

“lower-stepper-sensitivity” GtkSensitivityType

The sensitivity policy for the stepper that points to the adjustment's lower side.

Flags: Read / Write

Default value: GTK_SENSITIVITY_AUTO

The “restrict-to-fill-level” property

“restrict-to-fill-level” gboolean

The restrict-to-fill-level property controls whether slider movement is restricted to an upper boundary set by the fill level. See gtk_range_set_restrict_to_fill_level().

Flags: Read / Write

Default value: TRUE

Since: 2.12

The “round-digits” property

“round-digits” gint

The number of digits to round the value to when it changes, or -1. See “change-value”.

Flags: Read / Write

Allowed values: >= -1

Default value: -1

Since: 2.24

The “show-fill-level” property

“show-fill-level” gboolean

The show-fill-level property controls whether fill level indicator graphics are displayed on the trough. See gtk_range_set_show_fill_level().

Flags: Read / Write

Default value: FALSE

Since: 2.12

The “upper-stepper-sensitivity” property

“upper-stepper-sensitivity” GtkSensitivityType

The sensitivity policy for the stepper that points to the adjustment's upper side.

Flags: Read / Write

Default value: GTK_SENSITIVITY_AUTO

Style Property Details

The “arrow-displacement-x” style property

“arrow-displacement-x” gint

How far in the x direction to move the arrow when the button is depressed.

GtkRange:arrow-displacement-x has been deprecated since version 3.20 and should not be used in newly-written code.

The value of this style property is ignored.

Flags: Read

Default value: 0

The “arrow-displacement-y” style property

“arrow-displacement-y” gint

How far in the y direction to move the arrow when the button is depressed.

GtkRange:arrow-displacement-y has been deprecated since version 3.20 and should not be used in newly-written code.

The value of this style property is ignored.

Flags: Read

Default value: 0

The “arrow-scaling” style property

“arrow-scaling” gfloat

The arrow size proportion relative to the scroll button size.

GtkRange:arrow-scaling has been deprecated since version 3.20 and should not be used in newly-written code.

Use min-width/min-height on the "button" node instead. The value of this style property is ignored.

Flags: Read

Allowed values: [0,1]

Default value: 0.5

Since: 2.14

The “slider-width” style property

“slider-width” gint

Width of scrollbar or scale thumb.

GtkRange:slider-width has been deprecated since version 3.20 and should not be used in newly-written code.

Use the min-height/min-width CSS properties on the slider element. The value of this style property is ignored.

Flags: Read

Allowed values: >= 0

Default value: 14

The “stepper-size” style property

“stepper-size” gint

Length of step buttons at ends.

GtkRange:stepper-size has been deprecated since version 3.20 and should not be used in newly-written code.

Use the min-height/min-width CSS properties on the stepper elements. The value of this style property is ignored.

Flags: Read

Allowed values: >= 0

Default value: 14

The “stepper-spacing” style property

“stepper-spacing” gint

The spacing between the stepper buttons and thumb. Note that stepper-spacing won't have any effect if there are no steppers.

GtkRange:stepper-spacing has been deprecated since version 3.20 and should not be used in newly-written code.

Use the margin CSS property on the stepper elements. The value of this style property is ignored.

Flags: Read

Allowed values: >= 0

Default value: 0

The “trough-border” style property

```
  “trough-border”            gint
```

Spacing between thumb/steppers and outer trough bevel.

`GtkRange:trough-border` has been deprecated since version 3.20 and should not be used in newly-written code.

Use the margin/padding CSS properties on the trough and stepper elements. The value of this style property is ignored.

Flags: Read

Allowed values: >= 0

Default value: 1

The “trough-under-steppers” style property

```
  “trough-under-steppers”    gboolean
```

Whether to draw the trough across the full length of the range or to exclude the steppers and their spacing.

`GtkRange:trough-under-steppers` has been deprecated since version 3.20 and should not be used in newly-written code.

The value of this style property is ignored, and the widget will behave as if it was set to `TRUE`.

Flags: Read

Default value: TRUE

Since: 2.10

Signal Details

The “adjust-bounds” signal

```
void
user_function (GtkRange *range,
               gdouble   value,
               gpointer  user_data)
```

Emitted before clamping a value, to give the application a chance to adjust the bounds.

Parameters

range	the GtkRange that received the signal
value	the value before we clamp
user_data	user data set when the signal

	handler was connected.

Flags: Run Last

The “change-value” signal

```
gboolean
user_function (GtkRange     *range,
               GtkScrollType scroll,
               gdouble       value,
               gpointer      user_data)
```

The “change-value” signal is emitted when a scroll action is performed on a range. It allows an application to determine the type of scroll event that occurred and the resultant new value. The application can handle the event itself and return `TRUE` to prevent further processing. Or, by returning `FALSE`, it can pass the event to other handlers until the default GTK+ handler is reached.

The value parameter is unrounded. An application that overrides the GtkRange::change-value signal is responsible for clamping the value to the desired number of decimal digits; the default GTK+ handler clamps the value based on “round-digits”.

Parameters

range	the GtkRange that received the signal
scroll	the type of scroll action that was performed
value	the new value resulting from the scroll action
user_data	user data set when the signal handler was connected.

Returns

`TRUE` to prevent other handlers from being invoked for the signal, `FALSE` to propagate the signal further

Flags: Run Last

Since: 2.6

The “move-slider” signal

```
void
user_function (GtkRange     *range,
               GtkScrollType step,
               gpointer      user_data)
```

Virtual function that moves the slider. Used for keybindings.

Parameters

range	the GtkRange that received the signal
step	how to move the slider
user_data	user data set when the signal handler was connected.

Flags: Action

The `"value-changed"` signal

```
void
user_function (GtkRange *range,
               gpointer  user_data)
```

Emitted when the range value changes.

Parameters

range	the GtkRange that received the signal
user_data	user data set when the signal handler was connected.

Flags: Run Last

GtkIMContext

GtkIMContext — Base class for input method contexts

Functions

void	gtk_im_context_set_client_window ()
void	gtk_im_context_get_preedit_string ()
gboolean	gtk_im_context_filter_keypress ()
void	gtk_im_context_focus_in ()
void	gtk_im_context_focus_out ()
void	gtk_im_context_reset ()
void	gtk_im_context_set_cursor_location ()
void	gtk_im_context_set_use_preedit ()
void	gtk_im_context_set_surrounding ()
gboolean	gtk_im_context_get_surrounding ()
gboolean	gtk_im_context_delete_surrounding ()

Properties

GtkInputHints	input-hints	Read / Write
GtkInputPurpose	input-purpose	Read / Write

Signals

void	commit	Run Last
gboolean	delete-surrounding	Run Last
void	preedit-changed	Run Last
void	preedit-end	Run Last
void	preedit-start	Run Last
gboolean	retrieve-surrounding	Run Last

Types and Values

struct	GtkIMContext
struct	GtkIMContextClass
struct	GtkIMContextInfo

Object Hierarchy

```
GObject
└── GtkIMContext
    ├── GtkIMContextSimple
    └── GtkIMMulticontext
```

Includes

```
#include <gtk/gtk.h>
#include <gtk/gtkimmodule.h>
```

Description

GtkIMContext defines the interface for GTK+ input methods. An input method is used by GTK+ text input widgets like GtkEntry to map from key events to Unicode character strings.

The default input method can be set programmatically via the "gtk-im-module" GtkSettings property. Alternatively, you may set the GTK_IM_MODULE environment variable as documented in Running GTK+ Applications.

The GtkEntry "im-module" and GtkTextView "im-module" properties may also be used to set input methods for specific widget instances. For instance, a certain entry widget might be expected to contain certain characters which would be easier to input with a certain input method.

An input method may consume multiple key events in sequence and finally output the composed result. This is called preediting, and an input method may provide feedback about this process by displaying the intermediate composition states as preedit text. For instance, the default GTK+ input method implements the input of arbitrary Unicode code points by holding down the Control and Shift keys and then typing "U" followed by the hexadecimal digits of the code point. When releasing the Control and Shift keys, preediting ends and the character is inserted as text. Ctrl+Shift+u20AC for example results in the € sign.

Additional input methods can be made available for use by GTK+ widgets as loadable modules. An input method module is a small shared library which implements a subclass of GtkIMContext or GtkIMContextSimple and exports these four functions:

```
1                                               void im_module_init(GTypeModule *module);
```

This function should register the GType of the GtkIMContext subclass which implements the input method by means of `g_type_module_register_type()`. Note that `g_type_register_static()` cannot be used as the

type needs to be registered dynamically.

```
1                                                   void im_module_exit(void);
```

Here goes any cleanup code your input method might require on module unload.

```
1                                                   void im_module_list(const GtkIMContextInfo
2                                                   ***contexts, int *n_contexts)
3                                                   {
4                                                     *contexts = info_list;
5                                                     *n_contexts = G_N_ELEMENTS (info_list);
                                                    }
```

This function returns the list of input methods provided by the module. The example implementation above shows a common solution and simply returns a pointer to statically defined array of GtkIMContextInfo items for each provided input method.

```
1                                                   GtkIMContext * im_module_create(const gchar
                                                    *context_id);
```

This function should return a pointer to a newly created instance of the GtkIMContext subclass identified by `context_id` . The context ID is the same as specified in the GtkIMContextInfo array returned by `im_module_list()`.

After a new loadable input method module has been installed on the system, the configuration file `gtk.immodules` needs to be regenerated by gtk-query-immodules-3.0, in order for the new input method to become available to GTK+ applications.

Functions

gtk_im_context_set_client_window ()

```
void
gtk_im_context_set_client_window (GtkIMContext *context,
                                  GdkWindow *window);
```

Set the client window for the input context; this is the GdkWindow in which the input appears. This window is used in order to correctly position status windows, and may also be used for purposes internal to the input method.

Parameters

context	a GtkIMContext	
window	the client window. This may be `NULL` to indicate that the previous client window no longer exists.	[allow-none]

---

gtk_im_context_get_preedit_string ()

```
void
gtk_im_context_get_preedit_string (GtkIMContext *context,
                                   gchar **str,
                                   PangoAttrList **attrs,
                                   gint *cursor_pos);
```

Retrieve the current preedit string for the input context, and a list of attributes to apply to the string. This string should be displayed inserted at the insertion point.

Parameters

context	a GtkIMContext	
str	location to store the retrieved string. The string retrieved must be freed with `g_free()`.	[out][transfer full]
attrs	location to store the retrieved attribute list. When you are done with this list, you must unreference it with `pango_attr_list_unref()`.	[out][transfer full]
cursor_pos	location to store position of cursor (in characters) within the preedit string.	[out]

---

gtk_im_context_filter_keypress ()

```
gboolean
gtk_im_context_filter_keypress (GtkIMContext *context,
                                GdkEventKey *event);
```

Allow an input method to internally handle key press and release events. If this function returns `TRUE`, then no further processing should be done for this key event.

Parameters

context	a GtkIMContext
event	the key event

Returns

`TRUE` if the input method handled the key event.

---

gtk_im_context_focus_in ()

```
void
gtk_im_context_focus_in (GtkIMContext *context);
```

Notify the input method that the widget to which this input context corresponds has gained focus. The input method may, for example, change the displayed feedback to reflect this change.

Parameters

context	a GtkIMContext

---

gtk_im_context_focus_out ()

```
void
gtk_im_context_focus_out (GtkIMContext *context);
```

Notify the input method that the widget to which this input context corresponds has lost focus. The input method may, for example, change the displayed feedback or reset the contexts state to reflect this change.

Parameters

context	a GtkIMContext

---

gtk_im_context_reset ()

```
void
gtk_im_context_reset (GtkIMContext *context);
```

Notify the input method that a change such as a change in cursor position has been made. This will typically cause the input method to clear the preedit state.

Parameters

context	a GtkIMContext

---

gtk_im_context_set_cursor_location ()

```
void
gtk_im_context_set_cursor_location (GtkIMContext *context,
                                    const GdkRectangle *area);
```

Notify the input method that a change in cursor position has been made. The location is relative to the client window.

Parameters

context	a GtkIMContext
area	new location

---

gtk_im_context_set_use_preedit ()

```
void
gtk_im_context_set_use_preedit (GtkIMContext *context,
                                gboolean use_preedit);
```

Sets whether the IM context should use the preedit string to display feedback. If `use_preedit` is FALSE (default is TRUE), then the IM context may use some other method to display feedback, such as displaying it in a child of the root window.

Parameters

context	a GtkIMContext
use_preedit	whether the IM context should use the preedit string.

---

gtk_im_context_set_surrounding ()

```
void
gtk_im_context_set_surrounding (GtkIMContext *context,
                                const gchar *text,
                                gint len,
                                gint cursor_index);
```

Sets surrounding context around the insertion point and preedit string. This function is expected to be called in response to the GtkIMContext::retrieve_surrounding signal, and will likely have no effect if called at other times.

Parameters

context	a GtkIMContext
text	text surrounding the insertion point, as UTF-8. the preedit string should not be included within `text` .
len	the length of `text` , or -1 if `text` is nul-terminated
cursor_index	the byte index of the insertion cursor within `text` .

---

gtk_im_context_get_surrounding ()

```
gboolean
gtk_im_context_get_surrounding (GtkIMContext *context,
                                gchar **text,
                                gint *cursor_index);
```

Retrieves context around the insertion point. Input methods typically want context in order to constrain input text based on existing text; this is important for languages such as Thai where only some sequences of characters are allowed.

This function is implemented by emitting the GtkIMContext::retrieve_surrounding signal on the input method; in response to this signal, a widget should provide as much context as is available, up to an entire paragraph, by calling `gtk_im_context_set_surrounding()`. Note that there is no obligation for a widget to respond to the ::retrieve_surrounding signal, so input methods must be prepared to function without context.

Parameters

context	a GtkIMContext	
text	location to store a UTF-8 encoded string of text holding context around	[out][transfer full]

	the insertion point. If the function returns `TRUE`, then you must free the result stored in this location with `g_free()`.	
cursor_index	location to store byte index of the insertion cursor within `text` .	[out]

Returns

`TRUE` if surrounding text was provided; in this case you must free the result stored in *text.

---

gtk_im_context_delete_surrounding ()

```
gboolean
gtk_im_context_delete_surrounding (GtkIMContext *context,
                                   gint offset,
                                   gint n_chars);
```

Asks the widget that the input context is attached to to delete characters around the cursor position by emitting the GtkIMContext::delete_surrounding signal. Note that `offset` and `n_chars` are in characters not in bytes which differs from the usage other places in GtkIMContext.

In order to use this function, you should first call `gtk_im_context_get_surrounding()` to get the current context, and call this function immediately afterwards to make sure that you know what you are deleting. You should also account for the fact that even if the signal was handled, the input context might not have deleted all the characters that were requested to be deleted.

This function is used by an input method that wants to make subsitutions in the existing text in response to new input. It is not useful for applications.

Parameters

context	a GtkIMContext
offset	offset from cursor position in chars; a negative value means start before the cursor.
n_chars	number of characters to delete.

Returns

`TRUE` if the signal was handled.

Types and Values

struct GtkIMContext

```
struct GtkIMContext;
```

---

struct GtkIMContextClass

```
struct GtkIMContextClass {
  /* Signals */
  void     (*preedit_start)        (GtkIMContext *context);
  void     (*preedit_end)          (GtkIMContext *context);
  void     (*preedit_changed)      (GtkIMContext *context);
  void     (*commit)               (GtkIMContext *context, const gchar *str);
  gboolean (*retrieve_surrounding) (GtkIMContext *context);
  gboolean (*delete_surrounding)   (GtkIMContext *context,
                                    gint          offset,
                                    gint          n_chars);

  /* Virtual functions */
  void     (*set_client_window)   (GtkIMContext   *context,
                                   GdkWindow      *window);
  void     (*get_preedit_string)  (GtkIMContext   *context,
                                   gchar         **str,
                                   PangoAttrList **attrs,
                                   gint           *cursor_pos);
  gboolean (*filter_keypress)     (GtkIMContext   *context,
                                     GdkEventKey    *event);
  void     (*focus_in)            (GtkIMContext   *context);
  void     (*focus_out)           (GtkIMContext   *context);
  void     (*reset)               (GtkIMContext   *context);
  void     (*set_cursor_location) (GtkIMContext   *context,
                                   GdkRectangle   *area);
  void     (*set_use_preedit)     (GtkIMContext   *context,
                                   gboolean        use_preedit);
  void     (*set_surrounding)     (GtkIMContext   *context,
                                   const gchar    *text,
                                   gint            len,
                                   gint            cursor_index);
  gboolean (*get_surrounding)     (GtkIMContext   *context,
                                   gchar         **text,
                                   gint           *cursor_index);
};
```

Members

`preedit_start` ()	Default handler of the “preedit-start” signal.
`preedit_end` ()	Default handler of the “preedit-end” signal.
`preedit_changed` ()	Default handler of the “preedit-changed” signal.
`commit` ()	Default handler of the “commit” signal.
`retrieve_surrounding` ()	Default handler of the “retrieve-surrounding” signal.
`delete_surrounding` ()	Default handler of the “delete-surrounding” signal.
`set_client_window` ()	Called via `gtk_im_context_set_client_window()` when the input window where the entered text will appear

	changes. Override this to keep track of the current input window, for instance for the purpose of positioning a status display of your input method.
`get_preedit_string` ()	Called via `gtk_im_context_get_preedit_string()` to retrieve the text currently being preedited for display at the cursor position. Any input method which composes complex characters or any other compositions from multiple sequential key presses should override this method to provide feedback.
`filter_keypress` ()	Called via `gtk_im_context_filter_keypress()` on every key press or release event. Every non-trivial input method needs to override this in order to implement the mapping from key events to text. A return value of `TRUE` indicates to the caller that the event was consumed by the input method. In that case, the "commit" signal should be emitted upon completion of a key sequence to pass the resulting text back to the input widget. Alternatively, `FALSE` may be returned to indicate that the event wasn't handled by the input method. If a builtin mapping exists for the key, it is used to produce a character.
`focus_in` ()	Called via `gtk_im_context_focus_in()` when the input widget has gained focus. May be overridden to keep track of the current focus.
`focus_out` ()	Called via `gtk_im_context_focus_out()` when the input widget has lost focus. May be overridden to keep track of the current focus.
`reset` ()	Called via `gtk_im_context_reset()` to signal a change such as a change in cursor position. An input method that implements preediting should override this method to clear the preedit state on reset.
`set_cursor_location` ()	Called via

	gtk_im_context_set_cursor_location() to inform the input method of the current cursor location relative to the client window. May be overridden to implement the display of popup windows at the cursor position.
set_use_preedit ()	Called via gtk_im_context_set_use_preedit() to control the use of the preedit string. Override this to display feedback by some other means if turned off.
set_surrounding ()	Called via gtk_im_context_set_surrounding() in response to signal "retrieve-surrounding" to update the input method's idea of the context around the cursor. It is not necessary to override this method even with input methods which implement context-dependent behavior. The base implementation is sufficient for gtk_im_context_get_surrounding() to work.
get_surrounding ()	Called via gtk_im_context_get_surrounding() to update the context around the cursor location. It is not necessary to override this method even with input methods which implement context-dependent behavior. The base implementation emits "retrieve-surrounding" and records the context received by the subsequent invocation of get_surrounding .

---

struct GtkIMContextInfo

```
struct GtkIMContextInfo {
  const gchar *context_id;
  const gchar *context_name;
  const gchar *domain;
  const gchar *domain_dirname;
  const gchar *default_locales;
};
```

Bookkeeping information about a loadable input method.

Members

const gchar *`context_id`;	The unique identification string of the input method.
const gchar *`context_name`;	The human-readable name of the input method.
const gchar *`domain`;	Translation domain to be used with `dgettext()`
const gchar *`domain_dirname`;	Name of locale directory for use with `bindtextdomain()`
const gchar *`default_locales`;	A colon-separated list of locales where this input method should be the default. The asterisk “*” sets the default for all locales.

Property Details

The `“input-hints”` property

`“input-hints”` `GtkInputHints`

Hints for the text field behaviour.

Flags: Read / Write

The `“input-purpose”` property

`“input-purpose”` `GtkInputPurpose`

Purpose of the text field.

Flags: Read / Write

Default value: GTK_INPUT_PURPOSE_FREE_FORM

Signal Details

The `“commit”` signal

```
void
user_function (GtkIMContext *context,
               gchar        *str,
               gpointer      user_data)
```

The ::commit signal is emitted when a complete input sequence has been entered by the user. This can be a single character immediately after a key press or the final result of preediting.

Parameters

context	the object on which the signal is

	emitted
str	the completed character(s) entered by the user
user_data	user data set when the signal handler was connected.

Flags: Run Last

The “delete-surrounding” signal

```
gboolean
user_function (GtkIMContext *context,
               gint          offset,
               gint          n_chars,
               gpointer      user_data)
```

The ::delete-surrounding signal is emitted when the input method needs to delete all or part of the context surrounding the cursor.

Parameters

context	the object on which the signal is emitted
offset	the character offset from the cursor position of the text to be deleted. A negative value indicates a position before the cursor.
n_chars	the number of characters to be deleted
user_data	user data set when the signal handler was connected.

Returns

`TRUE` if the signal was handled.

Flags: Run Last

The “preedit-changed” signal

```
void
user_function (GtkIMContext *context,
               gpointer      user_data)
```

The ::preedit-changed signal is emitted whenever the preedit sequence currently being entered has changed. It is also emitted at the end of a preedit sequence, in which case `gtk_im_context_get_preedit_string()` returns the empty string.

Parameters

context	the object on which the signal is emitted
user_data	user data set when the signal handler was connected.

Flags: Run Last

The `"preedit-end"` signal

```
void
user_function (GtkIMContext *context,
               gpointer      user_data)
```

The ::preedit-end signal is emitted when a preediting sequence has been completed or canceled.

Parameters

context	the object on which the signal is emitted
user_data	user data set when the signal handler was connected.

Flags: Run Last

The `"preedit-start"` signal

```
void
user_function (GtkIMContext *context,
               gpointer      user_data)
```

The ::preedit-start signal is emitted when a new preediting sequence starts.

Parameters

context	the object on which the signal is emitted
user_data	user data set when the signal handler was connected.

Flags: Run Last

The `"retrieve-surrounding"` signal

```
gboolean
user_function (GtkIMContext *context,
               gpointer      user_data)
```

The ::retrieve-surrounding signal is emitted when the input method requires the context surrounding the cursor. The callback should set the input method surrounding context by calling the `gtk_im_context_set_surrounding()` method.

Parameters

context	the object on which the signal is emitted
user_data	user data set when the signal handler was connected.

Returns

TRUE if the signal was handled.

Flags: Run Last

GtkNativeDialog

GtkNativeDialog — Integrate with native dialogs

Functions

void	gtk_native_dialog_show ()
void	gtk_native_dialog_hide ()
void	gtk_native_dialog_destroy ()
gboolean	gtk_native_dialog_get_visible ()
void	gtk_native_dialog_set_modal ()
gboolean	gtk_native_dialog_get_modal ()
void	gtk_native_dialog_set_title ()
const char *	gtk_native_dialog_get_title ()
void	gtk_native_dialog_set_transient_for ()
GtkWindow *	gtk_native_dialog_get_transient_for ()
gint	gtk_native_dialog_run ()

Types and Values

#define	GTK_TYPE_NATIVE_DIALOG
struct	GtkNativeDialogClass

Includes

```
#include <gtk/gtk.h>
```

Description

Native dialogs are platform dialogs that don't use GtkDialog or GtkWindow. They are used in order to integrate better with a platform, by looking the same as other native applications and supporting platform specific features.

The GtkDialog functions cannot be used on such objects, but we need a similar API in order to drive them. The

GtkNativeDialog object is an API that allows you to do this. It allows you to set various common properties on the dialog, as well as show and hide it and get a “response” signal when the user finished with the dialog.

There is also a `gtk_native_dialog_run()` helper that makes it easy to run any native dialog in a modal way with a recursive mainloop, similar to `gtk_dialog_run()`.

Functions

gtk_native_dialog_show ()

```
void
gtk_native_dialog_show (GtkNativeDialog *self);
```

Shows the dialog on the display, allowing the user to interact with it. When the user accepts the state of the dialog the dialog will be automatically hidden and the “response” signal will be emitted.

Multiple calls while the dialog is visible will be ignored.

Parameters

self	a GtkNativeDialog

Since: 3.20

---

gtk_native_dialog_hide ()

```
void
gtk_native_dialog_hide (GtkNativeDialog *self);
```

Hides the dialog if it is visilbe, aborting any interaction. Once this is called the “response” signal will not be emitted until after the next call to `gtk_native_dialog_show()`.

If the dialog is not visible this does nothing.

Parameters

self	a GtkNativeDialog

Since: 3.20

---

gtk_native_dialog_destroy ()

```
void
gtk_native_dialog_destroy (GtkNativeDialog *self);
```

Destroys a dialog.

When a dialog is destroyed, it will break any references it holds to other objects. If it is visible it will be hidden and any underlying window system resources will be destroyed.

Note that this does not release any reference to the object (as opposed to destroying a GtkWindow) because

there is no reference from the windowing system to the GtkNativeDialog.

Parameters

self	a GtkNativeDialog

Since: 3.20

---

gtk_native_dialog_get_visible ()

```
gboolean
gtk_native_dialog_get_visible (GtkNativeDialog *self);
```

Determines whether the dialog is visible.

Parameters

self	a GtkNativeDialog

Returns

`TRUE` if the dialog is visible

Since: 3.20

---

gtk_native_dialog_set_modal ()

```
void
gtk_native_dialog_set_modal (GtkNativeDialog *self,
                             gboolean modal);
```

Sets a dialog modal or non-modal. Modal dialogs prevent interaction with other windows in the same application. To keep modal dialogs on top of main application windows, use `gtk_native_dialog_set_transient_for()` to make the dialog transient for the parent; most window managers will then disallow lowering the dialog below the parent.

Parameters

self	a GtkNativeDialog
modal	whether the window is modal

Since: 3.20

---

gtk_native_dialog_get_modal ()

```
gboolean
gtk_native_dialog_get_modal (GtkNativeDialog *self);
```

Returns whether the dialog is modal. See `gtk_native_dialog_set_modal()`.

Parameters

self	a GtkNativeDialog

Returns

`TRUE` if the dialog is set to be modal

Since: 3.20

---

gtk_native_dialog_set_title ()

```
void
gtk_native_dialog_set_title (GtkNativeDialog *self,
                             const char *title);
```

Sets the title of the GtkNativeDialog.

Parameters

self	a GtkNativeDialog
title	title of the dialog

Since: 3.20

---

gtk_native_dialog_get_title ()

```
const char *
gtk_native_dialog_get_title (GtkNativeDialog *self);
```

Gets the title of the GtkNativeDialog.

Parameters

self	a GtkNativeDialog

Returns

the title of the dialog, or `NULL` if none has been set explicitly. The returned string is owned by the widget and must not be modified or freed.

[nullable]

Since: 3.20

---

gtk_native_dialog_set_transient_for ()

```
void
gtk_native_dialog_set_transient_for (GtkNativeDialog *self,
                                     GtkWindow *parent);
```

Dialog windows should be set transient for the main application window they were spawned from. This allows window managers to e.g. keep the dialog on top of the main window, or center the dialog over the main window.

Passing `NULL` for `parent` unsets the current transient window.

Parameters

self	a GtkNativeDialog	
parent	parent window, or `NULL`.	[allow-none]

Since: 3.20

---

gtk_native_dialog_get_transient_for ()

```
GtkWindow *
gtk_native_dialog_get_transient_for (GtkNativeDialog *self);
```

Fetches the transient parent for this window. See `gtk_native_dialog_set_transient_for()`.

Parameters

self	a GtkNativeDialog

Returns

the transient parent for this window, or `NULL` if no transient parent has been set.

[nullable][transfer none]

Since: 3.20

---

gtk_native_dialog_run ()

```
gint
gtk_native_dialog_run (GtkNativeDialog *self);
```

Blocks in a recursive main loop until `self` emits the “response” signal. It then returns the response ID from the ::response signal emission.

Before entering the recursive main loop, `gtk_native_dialog_run()` calls `gtk_native_dialog_show()` on the dialog for you.

After `gtk_native_dialog_run()` returns, then dialog will be hidden.

Typical usage of this function might be:

```
1                                       gint result = gtk_native_dialog_run
```

```
2                                                   (GTK_NATIVE_DIALOG (dialog));
3                                                   switch (result)
4                                                     {
5                                                       case GTK_RESPONSE_ACCEPT:
6                                                          do_application_specific_something ();
7                                                          break;
8                                                       default:
9                                                          do_nothing_since_dialog_was_cancelled
10                                                  ();
11                                                         break;
                                                      }
                                                    g_object_unref (dialog);
```

Note that even though the recursive main loop gives the effect of a modal dialog (it prevents the user from interacting with other windows in the same window group while the dialog is run), callbacks such as timeouts, IO channel watches, DND drops, etc, will be triggered during a `gtk_nautilus_dialog_run()` call.

Parameters

self	a GtkNativeDialog

Returns

response ID

Since: 3.20

Types and Values

GTK_TYPE_NATIVE_DIALOG

```
#define GTK_TYPE_NATIVE_DIALOG             (gtk_native_dialog_get_type ())
```

---

struct GtkNativeDialogClass

```
struct GtkNativeDialogClass {
  GObjectClass parent_class;

  void (* response) (GtkNativeDialog *self, gint response_id);
};
```

See Also

GtkFileChooserNative, GtkDialog

Cross-process Embedding

GtkPlug — Toplevel for embedding into other processes

GtkSocket — Container for widgets from other processes

GtkPlug

GtkPlug — Toplevel for embedding into other processes

Functions

void	gtk_plug_construct ()
void	gtk_plug_construct_for_display ()
GtkWidget *	gtk_plug_new ()
GtkWidget *	gtk_plug_new_for_display ()
Window	gtk_plug_get_id ()
gboolean	gtk_plug_get_embedded ()
GdkWindow *	gtk_plug_get_socket_window ()

Properties

gboolean	embedded	Read
GdkWindow *	socket-window	Read

Signals

void	embedded	Run Last

Types and Values

struct	GtkPlug

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkContainer
            └── GtkBin
                └── GtkWindow
                    └── GtkPlug
```

Implemented Interfaces

GtkPlug implements AtkImplementorIface and GtkBuildable.

Includes

```
#include <gtk/gtkx.h>
```

Description

Together with GtkSocket, GtkPlug provides the ability to embed widgets from one process into another process in a fashion that is transparent to the user. One process creates a GtkSocket widget and passes the ID of that widget's window to the other process, which then creates a GtkPlug with that window ID. Any widgets contained in the GtkPlug then will appear inside the first application's window.

The communication between a GtkSocket and a GtkPlug follows the XEmbed Protocol. This protocol has also been implemented in other toolkits, e.g. Qt, allowing the same level of integration when embedding a Qt widget in GTK+ or vice versa.

The GtkPlug and GtkSocket widgets are only available when GTK+ is compiled for the X11 platform and `GDK_WINDOWING_X11` is defined. They can only be used on a GdkX11Display. To use GtkPlug and GtkSocket, you need to include the `gtk/gtkx.h` header.

Functions

gtk_plug_construct ()

```
void
gtk_plug_construct (GtkPlug *plug,
                    Window socket_id);
```

Finish the initialization of `plug` for a given GtkSocket identified by `socket_id` . This function will generally only be used by classes deriving from GtkPlug.

Parameters

plug	a GtkPlug.
socket_id	the XID of the socket's window.

---

gtk_plug_construct_for_display ()

```
void
gtk_plug_construct_for_display (GtkPlug *plug,
                                GdkDisplay *display,
                                Window socket_id);
```

Finish the initialization of `plug` for a given GtkSocket identified by `socket_id` which is currently displayed on `display` . This function will generally only be used by classes deriving from GtkPlug.

Parameters

plug	a GtkPlug.
display	the GdkDisplay associated with `socket_id` 's GtkSocket.
socket_id	the XID of the socket's window.

Since: 2.2

---

gtk_plug_new ()

```
GtkWidget *
gtk_plug_new (Window socket_id);
```

Creates a new plug widget inside the GtkSocket identified by `socket_id` . If `socket_id` is 0, the plug is left “unplugged” and can later be plugged into a GtkSocket by `gtk_socket_add_id()`.

Parameters

socket_id	the window ID of the socket, or 0.

Returns

the new GtkPlug widget.

---

gtk_plug_new_for_display ()

```
GtkWidget *
gtk_plug_new_for_display (GdkDisplay *display,
                          Window socket_id);
```

Create a new plug widget inside the GtkSocket identified by socket_id.

Parameters

display	the GdkDisplay on which `socket_id` is displayed
socket_id	the XID of the socket’s window.

Returns

the new GtkPlug widget.

Since: 2.2

---

gtk_plug_get_id ()

```
Window
gtk_plug_get_id (GtkPlug *plug);
```

Gets the window ID of a GtkPlug widget, which can then be used to embed this window inside another window, for instance with `gtk_socket_add_id()`.

Parameters

plug	a GtkPlug.

Returns

the window ID for the plug

---

gtk_plug_get_embedded ()

```
gboolean
gtk_plug_get_embedded (GtkPlug *plug);
```

Determines whether the plug is embedded in a socket.

Parameters

plug	a GtkPlug

Returns

`TRUE` if the plug is embedded in a socket

Since: 2.14

---

gtk_plug_get_socket_window ()

```
GdkWindow *
gtk_plug_get_socket_window (GtkPlug *plug);
```

Retrieves the socket the plug is embedded in.

Parameters

plug	a GtkPlug

Returns

the window of the socket, or `NULL`.

[nullable][transfer none]

Since: 2.14

Types and Values

struct GtkPlug

```
struct GtkPlug;
```

Property Details

The `“embedded”` property

```
  “embedded”                 gboolean
```

`TRUE` if the plug is embedded in a socket.

Flags: Read

Default value: FALSE

Since: 2.12

The `“socket-window”` property

```
  “socket-window”            GdkWindow *
```

The window of the socket the plug is embedded in.

Flags: Read

Since: 2.14

Signal Details

The `“embedded”` signal

```
void
user_function (GtkPlug *plug,
               gpointer user_data)
```

Gets emitted when the plug becomes embedded in a socket.

Parameters

plug	the object on which the signal was emitted
user_data	user data set when the signal handler was connected.

Flags: Run Last

See Also

GtkSocket

GtkSocket

GtkSocket — Container for widgets from other processes

Functions

GtkWidget *	gtk_socket_new ()
void	gtk_socket_add_id ()
Window	gtk_socket_get_id ()
GdkWindow *	gtk_socket_get_plug_window ()

Signals

void	plug-added	Run Last
gboolean	plug-removed	Run Last

Types and Values

struct	GtkSocket

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkContainer
            └── GtkSocket
```

Implemented Interfaces

GtkSocket implements AtkImplementorIface and GtkBuildable.

Includes

```
#include <gtk/gtkx.h>
```

Description

Together with GtkPlug, GtkSocket provides the ability to embed widgets from one process into another process in a fashion that is transparent to the user. One process creates a GtkSocket widget and passes that widget's window ID to the other process, which then creates a GtkPlug with that window ID. Any widgets contained in the GtkPlug then will appear inside the first application's window.

The socket's window ID is obtained by using `gtk_socket_get_id()`. Before using this function, the socket must have been realized, and for hence, have been added to its parent.

Obtaining the window ID of a socket.

```
GtkWidget *socket = gtk_socket_new ();
gtk_widget_show (socket);
gtk_container_add (GTK_CONTAINER (parent),
socket);

// The following call is only necessary if
one of
// the ancestors of the socket is not yet
visible.
gtk_widget_realize (socket);
g_print ("The ID of the sockets window is
%#x\n",
         gtk_socket_get_id (socket));
```

Note that if you pass the window ID of the socket to another process that will create a plug in the socket, you must make sure that the socket widget is not destroyed until that plug is created. Violating this rule will cause unpredictable consequences, the most likely consequence being that the plug will appear as a separate toplevel window. You can check if the plug has been created by using `gtk_socket_get_plug_window()`. If it returns a non-`NULL` value, then the plug has been successfully created inside of the socket.

When GTK+ is notified that the embedded window has been destroyed, then it will destroy the socket as well. You should always, therefore, be prepared for your sockets to be destroyed at any time when the main event loop is running. To prevent this from happening, you can connect to the "plug-removed" signal.

The communication between a GtkSocket and a GtkPlug follows the XEmbed Protocol. This protocol has also been implemented in other toolkits, e.g. Qt, allowing the same level of integration when embedding a Qt widget in GTK or vice versa.

The GtkPlug and GtkSocket widgets are only available when GTK+ is compiled for the X11 platform and `GDK_WINDOWING_X11` is defined. They can only be used on a GdkX11Display. To use GtkPlug and GtkSocket, you need to include the `gtk/gtkx.h` header.

Functions

gtk_socket_new ()

```
GtkWidget *
gtk_socket_new (void);
```

Create a new empty GtkSocket.

Returns

the new GtkSocket.

---

gtk_socket_add_id ()

```
void
gtk_socket_add_id (GtkSocket *socket_,
                   Window window);
```

Adds an XEMBED client, such as a GtkPlug, to the GtkSocket. The client may be in the same process or in a

different process.

To embed a GtkPlug in a GtkSocket, you can either create the GtkPlug with `gtk_plug_new (0)`, call `gtk_plug_get_id()` to get the window ID of the plug, and then pass that to the `gtk_socket_add_id()`, or you can call `gtk_socket_get_id()` to get the window ID for the socket, and call `gtk_plug_new()` passing in that ID.

The GtkSocket must have already be added into a toplevel window before you can make this call.

Parameters

socket_	a GtkSocket
window	the Window of a client participating in the XEMBED protocol.

---

gtk_socket_get_id ()

```
Window
gtk_socket_get_id (GtkSocket *socket_);
```

Gets the window ID of a GtkSocket widget, which can then be used to create a client embedded inside the socket, for instance with `gtk_plug_new()`.

The GtkSocket must have already be added into a toplevel window before you can make this call.

Parameters

socket_	a GtkSocket.

Returns

the window ID for the socket

---

gtk_socket_get_plug_window ()

```
GdkWindow *
gtk_socket_get_plug_window (GtkSocket *socket_);
```

Retrieves the window of the plug. Use this to check if the plug has been created inside of the socket.

Parameters

socket_	a GtkSocket.

Returns

the window of the plug if available, or `NULL`.

[nullable][transfer none]

Since: 2.14

Types and Values

struct GtkSocket

```
struct GtkSocket;
```

Signal Details

The “plug-added” signal

```
void
user_function (GtkSocket *socket_,
               gpointer   user_data)
```

This signal is emitted when a client is successfully added to the socket.

Parameters

socket_	the object which received the signal
user_data	user data set when the signal handler was connected.

Flags: Run Last

The “plug-removed” signal

```
gboolean
user_function (GtkSocket *socket_,
               gpointer   user_data)
```

This signal is emitted when a client is removed from the socket. The default action is to destroy the GtkSocket widget, so if you want to reuse it you must add a signal handler that returns `TRUE`.

Parameters

socket_	the object which received the signal
user_data	user data set when the signal handler was connected.

Returns

`TRUE` to stop other handlers from being invoked.

Flags: Run Last

See Also

GtkPlug, XEmbed Protocol

Recently Used Documents

GtkRecentManager — Managing recently used files

GtkRecentChooser — Interface implemented by widgets displaying recently used files

GtkRecentChooserDialog — Displays recently used files in a dialog

GtkRecentChooserMenu — Displays recently used files in a menu

GtkRecentChooserWidget — Displays recently used files

GtkRecentFilter — A filter for selecting a subset of recently used files

---

GtkRecentManager

GtkRecentManager — Managing recently used files

Functions

GtkRecentManager *	gtk_recent_manager_new ()
GtkRecentManager *	gtk_recent_manager_get_default ()
gboolean	gtk_recent_manager_add_item ()
gboolean	gtk_recent_manager_add_full ()
gboolean	gtk_recent_manager_remove_item ()
GtkRecentInfo *	gtk_recent_manager_lookup_item ()
gboolean	gtk_recent_manager_has_item ()
gboolean	gtk_recent_manager_move_item ()
GList *	gtk_recent_manager_get_items ()
gint	gtk_recent_manager_purge_items ()
GtkRecentInfo *	gtk_recent_info_ref ()
void	gtk_recent_info_unref ()
const gchar *	gtk_recent_info_get_uri ()
const gchar *	gtk_recent_info_get_display_name ()
const gchar *	gtk_recent_info_get_description ()
const gchar *	gtk_recent_info_get_mime_type ()
time_t	gtk_recent_info_get_added ()
time_t	gtk_recent_info_get_modified ()
time_t	gtk_recent_info_get_visited ()
gboolean	gtk_recent_info_get_private_hint ()
gboolean	gtk_recent_info_get_application_info ()
gchar **	gtk_recent_info_get_applications ()
gchar *	gtk_recent_info_last_application ()
gboolean	gtk_recent_info_has_application ()
GAppInfo *	gtk_recent_info_create_app_info ()
gchar **	gtk_recent_info_get_groups ()

gboolean	gtk_recent_info_has_group ()
GdkPixbuf *	gtk_recent_info_get_icon ()
GIcon *	gtk_recent_info_get_gicon ()
gchar *	gtk_recent_info_get_short_name ()
gchar *	gtk_recent_info_get_uri_display ()
gint	gtk_recent_info_get_age ()
gboolean	gtk_recent_info_is_local ()
gboolean	gtk_recent_info_exists ()
gboolean	gtk_recent_info_match ()

Properties

gchar *	filename	Read / Write / Construct Only
gint	size	Read

Signals

void	changed	Run First

Types and Values

struct	GtkRecentManager
	GtkRecentInfo
struct	GtkRecentData
#define	GTK_RECENT_MANAGER_ERROR
enum	GtkRecentManagerError

Object Hierarchy

```
GObject
└── GtkRecentManager
```

Includes

```
#include <gtk/gtk.h>
```

Description

GtkRecentManager provides a facility for adding, removing and looking up recently used files. Each recently used file is identified by its URI, and has meta-data associated to it, like the names and command lines of the applications that have registered it, the number of time each application has registered the same file, the mime type of the file and whether the file should be displayed only by the applications that have registered it.

The recently used files list is per user.

The GtkRecentManager acts like a database of all the recently used files. You can create new GtkRecentManager objects, but it is more efficient to use the default manager created by GTK+.

Adding a new recently used file is as simple as:

```
1                                   GtkRecentManager *manager;
2
```

```
3                                       manager = gtk_recent_manager_get_default ();
4                                       gtk_recent_manager_add_item (manager,
                                        file_uri);
```

The GtkRecentManager will try to gather all the needed information from the file itself through GIO.

Looking up the meta-data associated with a recently used file given its URI requires calling `gtk_recent_manager_lookup_item()`:

```
1                                       GtkRecentManager *manager;
2                                       GtkRecentInfo *info;
3                                       GError *error = NULL;
4
5                                       manager = gtk_recent_manager_get_default ();
6                                       info = gtk_recent_manager_lookup_item
7                                       (manager, file_uri, &error);
8                                       if (error)
9                                         {
10                                          g_warning ("Could not find the file: %s",
11                                      error->message);
12                                          g_error_free (error);
13                                        }
14                                      else
15                                       {
16                                         // Use the info object
                                           gtk_recent_info_unref (info);
                                        }
```

In order to retrieve the list of recently used files, you can use `gtk_recent_manager_get_items()`, which returns a list of GtkRecentInfo-structs.

A GtkRecentManager is the model used to populate the contents of one, or more GtkRecentChooser implementations.

Note that the maximum age of the recently used files list is controllable through the "gtk-recent-files-max-age" property.

Recently used files are supported since GTK+ 2.10.

Functions

gtk_recent_manager_new ()

```
GtkRecentManager *
gtk_recent_manager_new (void);
```

Creates a new recent manager object. Recent manager objects are used to handle the list of recently used resources. A GtkRecentManager object monitors the recently used resources list, and emits the "changed" signal each time something inside the list changes.

GtkRecentManager objects are expensive: be sure to create them only when needed. You should use `gtk_recent_manager_get_default()` instead.

Returns

A newly created GtkRecentManager object

Since: 2.10

---

gtk_recent_manager_get_default ()

```
GtkRecentManager *
gtk_recent_manager_get_default (void);
```

Gets a unique instance of GtkRecentManager, that you can share in your application without caring about memory management.

Returns

A unique GtkRecentManager. Do not ref or unref it.

[transfer none]

Since: 2.10

---

gtk_recent_manager_add_item ()

```
gboolean
gtk_recent_manager_add_item (GtkRecentManager *manager,
                             const gchar *uri);
```

Adds a new resource, pointed by `uri` , into the recently used resources list.

This function automatically retrieves some of the needed metadata and setting other metadata to common default values; it then feeds the data to `gtk_recent_manager_add_full()`.

See `gtk_recent_manager_add_full()` if you want to explicitly define the metadata for the resource pointed by `uri` .

Parameters

manager	a GtkRecentManager
uri	a valid URI

Returns

`TRUE` if the new item was successfully added to the recently used resources list

Since: 2.10

---

gtk_recent_manager_add_full ()

```
gboolean
gtk_recent_manager_add_full (GtkRecentManager *manager,
                             const gchar *uri,
                             const GtkRecentData *recent_data);
```

Adds a new resource, pointed by `uri` , into the recently used resources list, using the metadata specified inside the GtkRecentData passed in `recent_data` .

The passed URI will be used to identify this resource inside the list.

In order to register the new recently used resource, metadata about the resource must be passed as well as the URI; the metadata is stored in a GtkRecentData, which must contain the MIME type of the resource pointed by the URI; the name of the application that is registering the item, and a command line to be used when launching the item.

Optionally, a GtkRecentData might contain a UTF-8 string to be used when viewing the item instead of the last component of the URI; a short description of the item; whether the item should be considered private - that is, should be displayed only by the applications that have registered it.

Parameters

manager	a GtkRecentManager	
uri	a valid URI	
recent_data	metadata of the resource	

Returns

`TRUE` if the new item was successfully added to the recently used resources list, `FALSE` otherwise

Since: 2.10

---

gtk_recent_manager_remove_item ()

```
gboolean
gtk_recent_manager_remove_item (GtkRecentManager *manager,
                                const gchar *uri,
                                GError **error);
```

Removes a resource pointed by `uri` from the recently used resources list handled by a recent manager.

Parameters

manager	a GtkRecentManager	
uri	the URI of the item you wish to remove	
error	return location for a GError, or `NULL`.	[allow-none]

Returns

`TRUE` if the item pointed by `uri` has been successfully removed by the recently used resources list, and `FALSE` otherwise

Since: 2.10

---

gtk_recent_manager_lookup_item ()

```
GtkRecentInfo *
gtk_recent_manager_lookup_item (GtkRecentManager *manager,
                                const gchar *uri,
                                GError **error);
```

Searches for a URI inside the recently used resources list, and returns a GtkRecentInfo containing informations about the resource like its MIME type, or its display name.

Parameters

manager	a GtkRecentManager	
uri	a URI	
error	a return location for a GError, or `NULL`.	[allow-none]

Returns

a GtkRecentInfo containing information about the resource pointed by `uri` , or `NULL` if the URI was not registered in the recently used resources list. Free with `gtk_recent_info_unref()`.

[nullable]

Since: 2.10

---

gtk_recent_manager_has_item ()

```
gboolean
gtk_recent_manager_has_item (GtkRecentManager *manager,
                             const gchar *uri);
```

Checks whether there is a recently used resource registered with `uri` inside the recent manager.

Parameters

manager	a GtkRecentManager
uri	a URI

Returns

`TRUE` if the resource was found, `FALSE` otherwise

Since: 2.10

---

gtk_recent_manager_move_item ()

```
gboolean
gtk_recent_manager_move_item (GtkRecentManager *manager,
                              const gchar *uri,
```

```
                                const gchar *new_uri,
                                GError **error);
```

Changes the location of a recently used resource from `uri` to `new_uri`.

Please note that this function will not affect the resource pointed by the URIs, but only the URI used in the recently used resources list.

Parameters

manager	a GtkRecentManager	
uri	the URI of a recently used resource	
new_uri	the new URI of the recently used resource, or `NULL` to remove the item pointed by `uri` in the list.	[allow-none]
error	a return location for a GError, or `NULL`.	[allow-none]

Returns

`TRUE` on success

Since: 2.10

---

gtk_recent_manager_get_items ()

```
GList *
gtk_recent_manager_get_items (GtkRecentManager *manager);
```

Gets the list of recently used resources.

Parameters

manager	a GtkRecentManager

Returns

a list of newly allocated GtkRecentInfo objects. Use `gtk_recent_info_unref()` on each item inside the list, and then free the list itself using `g_list_free()`.

[element-type GtkRecentInfo][transfer full]

Since: 2.10

---

gtk_recent_manager_purge_items ()

```
gint
gtk_recent_manager_purge_items (GtkRecentManager *manager,
                                GError **error);
```

Purges every item from the recently used resources list.

Parameters

manager	a GtkRecentManager	
error	a return location for a GError, or NULL.	[allow-none]

Returns

the number of items that have been removed from the recently used resources list

Since: 2.10

---

gtk_recent_info_ref ()

```
GtkRecentInfo *
gtk_recent_info_ref (GtkRecentInfo *info);
```

Increases the reference count of `recent_info` by one.

Parameters

info	a GtkRecentInfo

Returns

the recent info object with its reference count increased by one

Since: 2.10

---

gtk_recent_info_unref ()

```
void
gtk_recent_info_unref (GtkRecentInfo *info);
```

Decreases the reference count of `info` by one. If the reference count reaches zero, `info` is deallocated, and the memory freed.

Parameters

info	a GtkRecentInfo

Since: 2.10

---

gtk_recent_info_get_uri ()

```
const gchar *
```

```
gtk_recent_info_get_uri (GtkRecentInfo *info);
```
Gets the URI of the resource.

Parameters

info	a GtkRecentInfo

Returns

the URI of the resource. The returned string is owned by the recent manager, and should not be freed.

Since: 2.10

---

gtk_recent_info_get_display_name ()

```
const gchar *
gtk_recent_info_get_display_name (GtkRecentInfo *info);
```
Gets the name of the resource. If none has been defined, the basename of the resource is obtained.

Parameters

info	a GtkRecentInfo

Returns

the display name of the resource. The returned string is owned by the recent manager, and should not be freed.

Since: 2.10

---

gtk_recent_info_get_description ()

```
const gchar *
gtk_recent_info_get_description (GtkRecentInfo *info);
```
Gets the (short) description of the resource.

Parameters

info	a GtkRecentInfo

Returns

the description of the resource. The returned string is owned by the recent manager, and should not be freed.

Since: 2.10

---

gtk_recent_info_get_mime_type ()

```
const gchar *
gtk_recent_info_get_mime_type (GtkRecentInfo *info);
```

Gets the MIME type of the resource.

Parameters

info	a GtkRecentInfo

Returns

the MIME type of the resource. The returned string is owned by the recent manager, and should not be freed.

Since: 2.10

---

gtk_recent_info_get_added ()

```
time_t
gtk_recent_info_get_added (GtkRecentInfo *info);
```

Gets the timestamp (seconds from system’s Epoch) when the resource was added to the recently used resources list.

Parameters

info	a GtkRecentInfo

Returns

the number of seconds elapsed from system’s Epoch when the resource was added to the list, or -1 on failure.

Since: 2.10

---

gtk_recent_info_get_modified ()

```
time_t
gtk_recent_info_get_modified (GtkRecentInfo *info);
```

Gets the timestamp (seconds from system’s Epoch) when the meta-data for the resource was last modified.

Parameters

info	a GtkRecentInfo

Returns

the number of seconds elapsed from system's Epoch when the resource was last modified, or -1 on failure.

Since: 2.10

---

gtk_recent_info_get_visited ()

```
time_t
gtk_recent_info_get_visited (GtkRecentInfo *info);
```

Gets the timestamp (seconds from system's Epoch) when the meta-data for the resource was last visited.

Parameters

info	a GtkRecentInfo

Returns

the number of seconds elapsed from system's Epoch when the resource was last visited, or -1 on failure.

Since: 2.10

---

gtk_recent_info_get_private_hint ()

```
gboolean
gtk_recent_info_get_private_hint (GtkRecentInfo *info);
```

Gets the value of the "private" flag. Resources in the recently used list that have this flag set to `TRUE` should only be displayed by the applications that have registered them.

Parameters

info	a GtkRecentInfo

Returns

`TRUE` if the private flag was found, `FALSE` otherwise

Since: 2.10

---

gtk_recent_info_get_application_info ()

```
gboolean
gtk_recent_info_get_application_info (GtkRecentInfo *info,
                                      const gchar *app_name,
                                      const gchar **app_exec,
                                      guint *count,
```

```
                                time_t *time_);
```

Gets the data regarding the application that has registered the resource pointed by `info` .

If the command line contains any escape characters defined inside the storage specification, they will be expanded.

Parameters

info	a GtkRecentInfo	
app_name	the name of the application that has registered this item	
app_exec	return location for the string containing the command line.	[transfer none][out]
count	return location for the number of times this item was registered.	[out]
time_	return location for the timestamp this item was last registered for this application.	[out]

Returns

`TRUE` if an application with `app_name` has registered this resource inside the recently used list, or `FALSE` otherwise. The `app_exec` string is owned by the GtkRecentInfo and should not be modified or freed

Since: 2.10

---

gtk_recent_info_get_applications ()

```
gchar **
gtk_recent_info_get_applications (GtkRecentInfo *info,
                                  gsize *length);
```

Retrieves the list of applications that have registered this resource.

Parameters

info	a GtkRecentInfo	
length	return location for the length of the returned list.	[out][allow-none]

Returns

a newly allocated `NULL`-terminated array of strings. Use `g_strfreev()` to free it.

[array length=length zero-terminated=1][transfer full]

Since: 2.10

---

gtk_recent_info_last_application ()

```
gchar *
gtk_recent_info_last_application (GtkRecentInfo *info);
```

Gets the name of the last application that have registered the recently used resource represented by `info`.

Parameters

info	a GtkRecentInfo	

Returns

an application name. Use `g_free()` to free it.

Since: 2.10

---

gtk_recent_info_has_application ()

```
gboolean
gtk_recent_info_has_application (GtkRecentInfo *info,
                                 const gchar *app_name);
```

Checks whether an application registered this resource using `app_name`.

Parameters

info	a GtkRecentInfo	
app_name	a string containing an application name	

Returns

`TRUE` if an application with name `app_name` was found, `FALSE` otherwise

Since: 2.10

---

gtk_recent_info_create_app_info ()

```
GAppInfo *
gtk_recent_info_create_app_info (GtkRecentInfo *info,
                                 const gchar *app_name,
                                 GError **error);
```

Creates a GAppInfo for the specified GtkRecentInfo

Parameters

info	a GtkRecentInfo	
app_name	the name of the application that	[allow-none]

	should be mapped to a GAppInfo; if `NULL` is used then the default application for the MIME type is used.	
error	return location for a GError, or `NULL`.	[allow-none]

Returns

the newly created GAppInfo, or `NULL`. In case of error, `error` will be set either with a `GTK_RECENT_MANAGER_ERROR` or a `G_IO_ERROR`.

[nullable][transfer full]

---

gtk_recent_info_get_groups ()

```
gchar **
gtk_recent_info_get_groups (GtkRecentInfo *info,
                            gsize *length);
```

Returns all groups registered for the recently used item `info` . The array of returned group names will be `NULL` terminated, so length might optionally be `NULL`.

Parameters

info	a GtkRecentInfo	
length	return location for the number of groups returned.	[out][allow-none]

Returns

a newly allocated `NULL` terminated array of strings. Use `g_strfreev()` to free it.

[array length=length zero-terminated=1][transfer full]

Since: 2.10

---

gtk_recent_info_has_group ()

```
gboolean
gtk_recent_info_has_group (GtkRecentInfo *info,
                           const gchar *group_name);
```

Checks whether `group_name` appears inside the groups registered for the recently used item `info` .

Parameters

info	a GtkRecentInfo
group_name	name of a group

Returns

`TRUE` if the group was found

Since: 2.10

---

gtk_recent_info_get_icon ()

```
GdkPixbuf *
gtk_recent_info_get_icon (GtkRecentInfo *info,
                          gint size);
```

Retrieves the icon of size `size` associated to the resource MIME type.

Parameters

info	a GtkRecentInfo
size	the size of the icon in pixels

Returns

a GdkPixbuf containing the icon, or `NULL`. Use `g_object_unref()` when finished using the icon.

[nullable][transfer full]

Since: 2.10

---

gtk_recent_info_get_gicon ()

```
GIcon *
gtk_recent_info_get_gicon (GtkRecentInfo *info);
```

Retrieves the icon associated to the resource MIME type.

Parameters

info	a GtkRecentInfo

Returns

a GIcon containing the icon, or `NULL`. Use `g_object_unref()` when finished using the icon.

[nullable][transfer full]

Since: 2.22

---

gtk_recent_info_get_short_name ()

```
gchar *
gtk_recent_info_get_short_name (GtkRecentInfo *info);
```

Computes a valid UTF-8 string that can be used as the name of the item in a menu or list. For example, calling this function on an item that refers to “file:///foo/bar.txt” will yield “bar.txt”.

Parameters

info	an GtkRecentInfo

Returns

A newly-allocated string in UTF-8 encoding free it with `g_free()`

Since: 2.10

---

gtk_recent_info_get_uri_display ()

```
gchar *
gtk_recent_info_get_uri_display (GtkRecentInfo *info);
```

Gets a displayable version of the resource’s URI. If the resource is local, it returns a local path; if the resource is not local, it returns the UTF-8 encoded content of `gtk_recent_info_get_uri()`.

Parameters

info	a GtkRecentInfo

Returns

a newly allocated UTF-8 string containing the resource’s URI or `NULL`. Use `g_free()` when done using it.

[nullable]

Since: 2.10

---

gtk_recent_info_get_age ()

```
gint
gtk_recent_info_get_age (GtkRecentInfo *info);
```

Gets the number of days elapsed since the last update of the resource pointed by `info` .

Parameters

info	a GtkRecentInfo

Returns

a positive integer containing the number of days elapsed since the time this resource was last modified

Since: 2.10

---

gtk_recent_info_is_local ()

```
gboolean
gtk_recent_info_is_local (GtkRecentInfo *info);
```

Checks whether the resource is local or not by looking at the scheme of its URI.

Parameters

info	a GtkRecentInfo

Returns

`TRUE` if the resource is local

Since: 2.10

---

gtk_recent_info_exists ()

```
gboolean
gtk_recent_info_exists (GtkRecentInfo *info);
```

Checks whether the resource pointed by `info` still exists. At the moment this check is done only on resources pointing to local files.

Parameters

info	a GtkRecentInfo

Returns

`TRUE` if the resource exists

Since: 2.10

---

gtk_recent_info_match ()

```
gboolean
gtk_recent_info_match (GtkRecentInfo *info_a,
                       GtkRecentInfo *info_b);
```

Checks whether two GtkRecentInfo point to the same resource.

Parameters

info_a	a GtkRecentInfo
info_b	a GtkRecentInfo

Returns

`TRUE` if both GtkRecentInfo point to the same resource, `FALSE` otherwise

Since: 2.10

Types and Values

struct GtkRecentManager

```
struct GtkRecentManager;
```

GtkRecentManager contains only private data and should be accessed using the provided API.

Since: 2.10

GtkRecentInfo

```
typedef struct _GtkRecentInfo GtkRecentInfo;
```

GtkRecentInfo contains private data only, and should be accessed using the provided API.

GtkRecentInfo constains all the meta-data associated with an entry in the recently used files list.

Since: 2.10

---

struct GtkRecentData

```
struct GtkRecentData {
  gchar *display_name;
  gchar *description;

  gchar *mime_type;

  gchar *app_name;
  gchar *app_exec;

  gchar **groups;

  gboolean is_private;
};
```

Meta-data to be passed to `gtk_recent_manager_add_full()` when registering a recently used resource.

Members

gchar *`display_name`;	a UTF-8 encoded string, containing the name of the recently used resource to be displayed, or `NULL`;	
gchar *`description`;	a UTF-8 encoded string, containing a short description of the resource, or `NULL`;	
gchar *`mime_type`;	the MIME type of the resource;	
gchar *`app_name`;	the name of the application that is registering this recently used resource;	
gchar *`app_exec`;	command line used to launch this resource; may contain the “%f” and “%u” escape characters which will be expanded to the resource file path and URI respectively when the command line is retrieved;	
gchar **`groups`;	a vector of strings containing groups names;.	[array zero-terminated=1]
gboolean `is_private`;	whether this resource should be displayed only by the applications that have registered it or not.	

GTK_RECENT_MANAGER_ERROR

```
#define GTK_RECENT_MANAGER_ERROR (gtk_recent_manager_error_quark ())
```

The GError domain for GtkRecentManager errors.

Since: 2.10

---

enum GtkRecentManagerError

Error codes for GtkRecentManager operations

Members

GTK_RECENT_MANAGER_ERROR_NOT_FOUND	the URI specified does not exists in the recently used resources list.
GTK_RECENT_MANAGER_ERROR_INVALID_URI	the URI specified is not valid.
GTK_RECENT_MANAGER_ERROR_INVALID_ENCODING	the supplied string is not UTF-8 encoded.
GTK_RECENT_MANAGER_ERROR_NOT_REGISTERED	no application has registered the specified item.
GTK_RECENT_MANAGER_ERROR_READ	failure while reading the recently used resources file.
GTK_RECENT_MANAGER_ERR	failure while writing the recently

OR_WRITE	used resources file.
GTK_RECENT_MANAGER_ERR OR_UNKNOWN	unspecified error.

Since: 2.10

Property Details

The `“filename”` property

```
  “filename”                  gchar *
```

The full path to the file to be used to store and read the recently used resources list

Flags: Read / Write / Construct Only

Default value: NULL

Since: 2.10

The `“size”` property

```
  “size”                      gint
```

The size of the recently used resources list.

Flags: Read

Allowed values: >= -1

Default value: 0

Since: 2.10

Signal Details

The `“changed”` signal

```
void
user_function (GtkRecentManager *recent_manager,
               gpointer          user_data)
```

Emitted when the current recently used resources manager changes its contents, either by calling `gtk_recent_manager_add_item()` or by another application.

Parameters

recent_manager	the recent manager
user_data	user data set when the signal handler was connected.

Flags: Run First

Since: 2.10

See Also

GBookmarkFile, GtkSettings, GtkRecentChooser

GtkRecentChooser

GtkRecentChooser — Interface implemented by widgets displaying recently used files

Functions

void	gtk_recent_chooser_set_show_private ()
gboolean	gtk_recent_chooser_get_show_private ()
void	gtk_recent_chooser_set_show_not_found ()
gboolean	gtk_recent_chooser_get_show_not_found ()
void	gtk_recent_chooser_set_show_icons ()
gboolean	gtk_recent_chooser_get_show_icons ()
void	gtk_recent_chooser_set_select_multiple ()
gboolean	gtk_recent_chooser_get_select_multiple ()
void	gtk_recent_chooser_set_local_only ()
gboolean	gtk_recent_chooser_get_local_only ()
void	gtk_recent_chooser_set_limit ()
gint	gtk_recent_chooser_get_limit ()
void	gtk_recent_chooser_set_show_tips ()
gboolean	gtk_recent_chooser_get_show_tips ()
void	gtk_recent_chooser_set_sort_type ()
GtkRecentSortType	gtk_recent_chooser_get_sort_type ()
gint	(*GtkRecentSortFunc) ()
void	gtk_recent_chooser_set_sort_func ()
gboolean	gtk_recent_chooser_set_current_uri ()
gchar *	gtk_recent_chooser_get_current_uri ()
GtkRecentInfo *	gtk_recent_chooser_get_current_item ()
gboolean	gtk_recent_chooser_select_uri ()
void	gtk_recent_chooser_unselect_uri ()
void	gtk_recent_chooser_select_all ()
void	gtk_recent_chooser_unselect_all ()
GList *	gtk_recent_chooser_get_items ()
gchar **	gtk_recent_chooser_get_uris ()
void	gtk_recent_chooser_add_filter ()
void	gtk_recent_chooser_remove_filter ()
GSList *	gtk_recent_chooser_list_filters ()
void	gtk_recent_chooser_set_filter ()
GtkRecentFilter *	gtk_recent_chooser_get_filter ()

Properties

GtkRecentFilter *	filter	Read / Write
gint	limit	Read / Write
gboolean	local-only	Read / Write
GtkRecentManager *	recent-manager	Write / Construct Only
gboolean	select-multiple	Read / Write
gboolean	show-icons	Read / Write
gboolean	show-not-found	Read / Write
gboolean	show-private	Read / Write
gboolean	show-tips	Read / Write
GtkRecentSortType	sort-type	Read / Write

Signals

void	item-activated	Run Last
void	selection-changed	Run Last

Types and Values

	GtkRecentChooser
struct	GtkRecentChooserIface
#define	GTK_RECENT_CHOOSER_ERROR
enum	GtkRecentChooserError
enum	GtkRecentSortType

Object Hierarchy

```
GInterface
└── GtkRecentChooser
```

Prerequisites

GtkRecentChooser requires GObject.

Known Implementations

GtkRecentChooser is implemented by GtkRecentAction, GtkRecentChooserDialog, GtkRecentChooserMenu and GtkRecentChooserWidget.

Includes

```
#include <gtk/gtk.h>
```

Description

GtkRecentChooser is an interface that can be implemented by widgets displaying the list of recently used files. In GTK+, the main objects that implement this interface are GtkRecentChooserWidget,

GtkRecentChooserDialog and GtkRecentChooserMenu.

Recently used files are supported since GTK+ 2.10.

Functions

gtk_recent_chooser_set_show_private ()

```
void
gtk_recent_chooser_set_show_private (GtkRecentChooser *chooser,
                                     gboolean show_private);
```

Whether to show recently used resources marked registered as private.

Parameters

chooser	a GtkRecentChooser
show_private	TRUE to show private items, FALSE otherwise

Since: 2.10

---

gtk_recent_chooser_get_show_private ()

```
gboolean
gtk_recent_chooser_get_show_private (GtkRecentChooser *chooser);
```

Returns whether chooser should display recently used resources registered as private.

Parameters

chooser	a GtkRecentChooser

Returns

TRUE if the recent chooser should show private items, FALSE otherwise.

Since: 2.10

---

gtk_recent_chooser_set_show_not_found ()

```
void
gtk_recent_chooser_set_show_not_found (GtkRecentChooser *chooser,
                                       gboolean show_not_found);
```

Sets whether chooser should display the recently used resources that it didn’t find. This only applies to local resources.

Parameters

chooser	a GtkRecentChooser
show_not_found	whether to show the local items we didn’t find

Since: 2.10

---

gtk_recent_chooser_get_show_not_found ()

```
gboolean
gtk_recent_chooser_get_show_not_found (GtkRecentChooser *chooser);
```

Retrieves whether `chooser` should show the recently used resources that were not found.

Parameters

chooser	a GtkRecentChooser

Returns

`TRUE` if the resources not found should be displayed, and `FALSE` otheriwse.

Since: 2.10

---

gtk_recent_chooser_set_show_icons ()

```
void
gtk_recent_chooser_set_show_icons (GtkRecentChooser *chooser,
                                   gboolean show_icons);
```

Sets whether `chooser` should show an icon near the resource when displaying it.

Parameters

chooser	a GtkRecentChooser
show_icons	whether to show an icon near the resource

Since: 2.10

---

gtk_recent_chooser_get_show_icons ()

```
gboolean
gtk_recent_chooser_get_show_icons (GtkRecentChooser *chooser);
```

Retrieves whether `chooser` should show an icon near the resource.

Parameters

chooser	a GtkRecentChooser

Returns

TRUE if the icons should be displayed, FALSE otherwise.

Since: 2.10

---

gtk_recent_chooser_set_select_multiple ()

```
void
gtk_recent_chooser_set_select_multiple
                                (GtkRecentChooser *chooser,
                                 gboolean select_multiple);
```

Sets whether chooser can select multiple items.

Parameters

chooser	a GtkRecentChooser
select_multiple	TRUE if chooser can select more than one item

Since: 2.10

---

gtk_recent_chooser_get_select_multiple ()

```
gboolean
gtk_recent_chooser_get_select_multiple
                                (GtkRecentChooser *chooser);
```

Gets whether chooser can select multiple items.

Parameters

chooser	a GtkRecentChooser

Returns

TRUE if chooser can select more than one item.

Since: 2.10

---

gtk_recent_chooser_set_local_only ()

```
void
gtk_recent_chooser_set_local_only (GtkRecentChooser *chooser,
```

```
                                gboolean local_only);
```

Sets whether only local resources, that is resources using the file:// URI scheme, should be shown in the recently used resources selector. If `local_only` is `TRUE` (the default) then the shown resources are guaranteed to be accessible through the operating system native file system.

Parameters

chooser	a GtkRecentChooser
local_only	`TRUE` if only local files can be shown

Since: 2.10

---

gtk_recent_chooser_get_local_only ()

```
gboolean
gtk_recent_chooser_get_local_only (GtkRecentChooser *chooser);
```

Gets whether only local resources should be shown in the recently used resources selector. See `gtk_recent_chooser_set_local_only()`

Parameters

chooser	a GtkRecentChooser

Returns

`TRUE` if only local resources should be shown.

Since: 2.10

---

gtk_recent_chooser_set_limit ()

```
void
gtk_recent_chooser_set_limit (GtkRecentChooser *chooser,
                              gint limit);
```

Sets the number of items that should be returned by `gtk_recent_chooser_get_items()` and `gtk_recent_chooser_get_uris()`.

Parameters

chooser	a GtkRecentChooser
limit	a positive integer, or -1 for all items

Since: 2.10

---

gtk_recent_chooser_get_limit ()

```
gint
gtk_recent_chooser_get_limit (GtkRecentChooser *chooser);
```

Gets the number of items returned by `gtk_recent_chooser_get_items()` and `gtk_recent_chooser_get_uris()`.

Parameters

chooser	a GtkRecentChooser

Returns

A positive integer, or -1 meaning that all items are returned.

Since: 2.10

---

gtk_recent_chooser_set_show_tips ()

```
void
gtk_recent_chooser_set_show_tips (GtkRecentChooser *chooser,
                                  gboolean show_tips);
```

Sets whether to show a tooltips containing the full path of each recently used resource in a GtkRecentChooser widget.

Parameters

chooser	a GtkRecentChooser
show_tips	`TRUE` if tooltips should be shown

Since: 2.10

---

gtk_recent_chooser_get_show_tips ()

```
gboolean
gtk_recent_chooser_get_show_tips (GtkRecentChooser *chooser);
```

Gets whether `chooser` should display tooltips containing the full path of a recently user resource.

Parameters

chooser	a GtkRecentChooser

Returns

`TRUE` if the recent chooser should show tooltips, `FALSE` otherwise.

Since: 2.10

---

gtk_recent_chooser_set_sort_type ()

```
void
gtk_recent_chooser_set_sort_type (GtkRecentChooser *chooser,
                                  GtkRecentSortType sort_type);
```

Changes the sorting order of the recently used resources list displayed by `chooser` .

Parameters

chooser	a GtkRecentChooser
sort_type	sort order that the chooser should use

Since: 2.10

---

gtk_recent_chooser_get_sort_type ()

```
GtkRecentSortType
gtk_recent_chooser_get_sort_type (GtkRecentChooser *chooser);
```

Gets the value set by `gtk_recent_chooser_set_sort_type()`.

Parameters

chooser	a GtkRecentChooser

Returns

the sorting order of the `chooser` .

Since: 2.10

GtkRecentSortFunc ()

```
gint
(*GtkRecentSortFunc) (GtkRecentInfo *a,
                      GtkRecentInfo *b,
                      gpointer user_data);
```

---

gtk_recent_chooser_set_sort_func ()

```
void
gtk_recent_chooser_set_sort_func (GtkRecentChooser *chooser,
                                  GtkRecentSortFunc sort_func,
                                  gpointer sort_data,
                                  GDestroyNotify data_destroy);
```

Sets the comparison function used when sorting to be `sort_func` . If the `chooser` has the sort type set to

GTK_RECENT_SORT_CUSTOM then the chooser will sort using this function.

To the comparison function will be passed two GtkRecentInfo structs and `sort_data` ; `sort_func` should return a positive integer if the first item comes before the second, zero if the two items are equal and a negative integer if the first item comes after the second.

Parameters

chooser	a GtkRecentChooser	
sort_func	the comparison function	
sort_data	user data to pass to `sort_func` , or `NULL`.	[allow-none]
data_destroy	destroy notifier for `sort_data` , or `NULL`.	[allow-none]

Since: 2.10

---

gtk_recent_chooser_set_current_uri ()

```
gboolean
gtk_recent_chooser_set_current_uri (GtkRecentChooser *chooser,
                                    const gchar *uri,
                                    GError **error);
```

Sets `uri` as the current URI for `chooser` .

Parameters

chooser	a GtkRecentChooser	
uri	a URI	
error	return location for a GError, or `NULL`.	[allow-none]

Returns

`TRUE` if the URI was found.

Since: 2.10

---

gtk_recent_chooser_get_current_uri ()

```
gchar *
gtk_recent_chooser_get_current_uri (GtkRecentChooser *chooser);
```

Gets the URI currently selected by `chooser` .

Parameters

chooser	a GtkRecentChooser	

Returns

a newly allocated string holding a URI.

Since: 2.10

---

gtk_recent_chooser_get_current_item ()

```
GtkRecentInfo *
gtk_recent_chooser_get_current_item (GtkRecentChooser *chooser);
```

Gets the GtkRecentInfo currently selected by `chooser` .

Parameters

chooser	a GtkRecentChooser	

Returns

a GtkRecentInfo. Use `gtk_recent_info_unref()` when when you have finished using it.

Since: 2.10

---

gtk_recent_chooser_select_uri ()

```
gboolean
gtk_recent_chooser_select_uri (GtkRecentChooser *chooser,
                               const gchar *uri,
                               GError **error);
```

Selects `uri` inside `chooser` .

Parameters

chooser	a GtkRecentChooser	
uri	a URI	
error	return location for a GError, or `NULL`.	[allow-none]

Returns

`TRUE` if `uri` was found.

Since: 2.10

---

gtk_recent_chooser_unselect_uri ()

```
void
gtk_recent_chooser_unselect_uri (GtkRecentChooser *chooser,
                                 const gchar *uri);
```

Unselects `uri` inside `chooser` .

Parameters

chooser	a GtkRecentChooser
uri	a URI

Since: 2.10

---

gtk_recent_chooser_select_all ()

```
void
gtk_recent_chooser_select_all (GtkRecentChooser *chooser);
```

Selects all the items inside `chooser` , if the `chooser` supports multiple selection.

Parameters

chooser	a GtkRecentChooser

Since: 2.10

---

gtk_recent_chooser_unselect_all ()

```
void
gtk_recent_chooser_unselect_all (GtkRecentChooser *chooser);
```

Unselects all the items inside `chooser` .

Parameters

chooser	a GtkRecentChooser

Since: 2.10

---

gtk_recent_chooser_get_items ()

```
GList *
gtk_recent_chooser_get_items (GtkRecentChooser *chooser);
```

Gets the list of recently used resources in form of GtkRecentInfo objects.

The return value of this function is affected by the “sort-type” and “limit” properties of `chooser` .

Parameters

chooser	a GtkRecentChooser

Returns

A newly allocated list of GtkRecentInfo objects. You should use `gtk_recent_info_unref()` on every item of the list, and then free the list itself using `g_list_free()`.

[element-type GtkRecentInfo][transfer full]

Since: 2.10

---

gtk_recent_chooser_get_uris ()

```
gchar **
gtk_recent_chooser_get_uris (GtkRecentChooser *chooser,
                             gsize *length);
```

Gets the URI of the recently used resources.

The return value of this function is affected by the “sort-type” and “limit” properties of `chooser` .

Since the returned array is `NULL` terminated, `length` may be `NULL`.

Parameters

chooser	a GtkRecentChooser	
length	return location for a the length of the URI list, or `NULL`.	[out][allow-none]

Returns

A newly allocated, `NULL`-terminated array of strings. Use `g_strfreev()` to free it.

[array length=length zero-terminated=1][transfer full]

Since: 2.10

---

gtk_recent_chooser_add_filter ()

```
void
gtk_recent_chooser_add_filter (GtkRecentChooser *chooser,
                               GtkRecentFilter *filter);
```

Adds `filter` to the list of GtkRecentFilter objects held by `chooser` .

If no previous filter objects were defined, this function will call `gtk_recent_chooser_set_filter()`.

Parameters

chooser	a GtkRecentChooser
filter	a GtkRecentFilter

Since: 2.10

---

gtk_recent_chooser_remove_filter ()

```
void
gtk_recent_chooser_remove_filter (GtkRecentChooser *chooser,
                                  GtkRecentFilter *filter);
```

Removes `filter` from the list of GtkRecentFilter objects held by `chooser` .

Parameters

chooser	a GtkRecentChooser
filter	a GtkRecentFilter

Since: 2.10

---

gtk_recent_chooser_list_filters ()

```
GSList *
gtk_recent_chooser_list_filters (GtkRecentChooser *chooser);
```

Gets the GtkRecentFilter objects held by `chooser` .

Parameters

chooser	a GtkRecentChooser

Returns

A singly linked list of GtkRecentFilter objects. You should just free the returned list using `g_slist_free()`.

[element-type GtkRecentFilter][transfer container]

Since: 2.10

---

gtk_recent_chooser_set_filter ()

```
void
gtk_recent_chooser_set_filter (GtkRecentChooser *chooser,
                               GtkRecentFilter *filter);
```

Sets `filter` as the current GtkRecentFilter object used by `chooser` to affect the displayed recently used resources.

Parameters

chooser	a GtkRecentChooser	
filter	a GtkRecentFilter.	[allow-none]

Since: 2.10

---

gtk_recent_chooser_get_filter ()

```
GtkRecentFilter *
gtk_recent_chooser_get_filter (GtkRecentChooser *chooser);
```

Gets the GtkRecentFilter object currently used by `chooser` to affect the display of the recently used resources.

Parameters

chooser	a GtkRecentChooser

Returns

a GtkRecentFilter object.

[transfer none]

Since: 2.10

Types and Values

GtkRecentChooser

```
typedef struct _GtkRecentChooser GtkRecentChooser;
```

---

struct GtkRecentChooserIface

```
struct GtkRecentChooserIface {
  /*
   * Methods
   */
  gboolean          (* set_current_uri)    (GtkRecentChooser  *chooser,
                                            const gchar       *uri,
                                            GError           **error);
  gchar *           (* get_current_uri)    (GtkRecentChooser  *chooser);
  gboolean          (* select_uri)         (GtkRecentChooser  *chooser,
                                            const gchar       *uri,
                                            GError           **error);
  void              (* unselect_uri)       (GtkRecentChooser  *chooser,
                                             const gchar       *uri);
  void              (* select_all)         (GtkRecentChooser  *chooser);
  void              (* unselect_all)       (GtkRecentChooser  *chooser);
  GList *           (* get_items)          (GtkRecentChooser  *chooser);
  GtkRecentManager *(* get_recent_manager) (GtkRecentChooser  *chooser);
```

```
  void                (* add_filter)         (GtkRecentChooser  *chooser,
                                              GtkRecentFilter   *filter);
  void                (* remove_filter)      (GtkRecentChooser  *chooser,
                                              GtkRecentFilter   *filter);
  GSList *            (* list_filters)       (GtkRecentChooser  *chooser);
  void                (* set_sort_func)      (GtkRecentChooser  *chooser,
                                              GtkRecentSortFunc  sort_func,
                                              gpointer           sort_data,
                                              GDestroyNotify     data_destroy);

  /*
   * Signals
   */
  void                 (* item_activated)     (GtkRecentChooser  *chooser);
  void                 (* selection_changed)  (GtkRecentChooser  *chooser);
};
```

Members

`set_current_uri` ()	Sets uri as the current URI for chooser.
`get_current_uri` ()	Gets the URI currently selected by chooser.
`select_uri` ()	Selects uri inside chooser.
`unselect_uri` ()	Unselects uri inside chooser.
`select_all` ()	Selects all the items inside chooser, if the chooser supports multiple selection.
`unselect_all` ()	Unselects all the items inside chooser.
`get_items` ()	Gets the list of recently used resources in form of GtkRecentInfo objects.
`get_recent_manager` ()	Gets the GtkRecentManager used by chooser.
`add_filter` ()	Adds filter to the list of GtkRecentFilter objects held by chooser.
`remove_filter` ()	Removes filter from the list of GtkRecentFilter objects held by chooser.
`list_filters` ()	Gets the GtkRecentFilter objects held by chooser.
`set_sort_func` ()	Sets the comparison function used when sorting to be sort_func.
`item_activated` ()	Signal emitted when the user “activates” a recent item in the recent chooser.
`selection_changed` ()	Signal emitted when there is a change in the set of selected recently used resources.

GTK_RECENT_CHOOSER_ERROR

```
#define GTK_RECENT_CHOOSER_ERROR (gtk_recent_chooser_error_quark ())
```

Used to get the GError quark for GtkRecentChooser errors.

Since: 2.10

---

enum GtkRecentChooserError

These identify the various errors that can occur while calling GtkRecentChooser functions.

Members

GTK_RECENT_CHOOSER_ERROR_NOT_FOUND	Indicates that a file does not exist
GTK_RECENT_CHOOSER_ERROR_INVALID_URI	Indicates a malformed URI

Since: 2.10

---

enum GtkRecentSortType

Used to specify the sorting method to be applyed to the recently used resource list.

Members

GTK_RECENT_SORT_NONE	Do not sort the returned list of recently used resources.
GTK_RECENT_SORT_MRU	Sort the returned list with the most recently used items first.
GTK_RECENT_SORT_LRU	Sort the returned list with the least recently used items first.
GTK_RECENT_SORT_CUSTOM	Sort the returned list using a custom sorting function passed using `gtk_recent_chooser_set_sort_func()`.

Since: 2.10

Property Details

The `"filter"` property

```
  "filter"                   GtkRecentFilter *
```

The GtkRecentFilter object to be used when displaying the recently used resources.

Flags: Read / Write

Since: 2.10

The “limit” property

“limit” gint

The maximum number of recently used resources to be displayed, or -1 to display all items.

Flags: Read / Write

Allowed values: >= -1

Default value: 50

Since: 2.10

The “local-only” property

“local-only” gboolean

Whether this GtkRecentChooser should display only local (file:) resources.

Flags: Read / Write

Default value: TRUE

Since: 2.10

The “recent-manager” property

“recent-manager” GtkRecentManager *

The GtkRecentManager instance used by the GtkRecentChooser to display the list of recently used resources.

Flags: Write / Construct Only

Since: 2.10

The “select-multiple” property

“select-multiple” gboolean

Allow the user to select multiple resources.

Flags: Read / Write

Default value: FALSE

Since: 2.10

The “show-icons” property

“show-icons” gboolean

Whether this GtkRecentChooser should display an icon near the item.

Flags: Read / Write

Default value: TRUE

Since: 2.10

The “show-not-found” property

“show-not-found” gboolean

Whether this GtkRecentChooser should display the recently used resources even if not present anymore. Setting this to FALSE will perform a potentially expensive check on every local resource (every remote resource will always be displayed).

Flags: Read / Write

Default value: TRUE

Since: 2.10

The “show-private” property

“show-private” gboolean

Whether the private items should be displayed.

Flags: Read / Write

Default value: FALSE

The “show-tips” property

“show-tips” gboolean

Whether this GtkRecentChooser should display a tooltip containing the full path of the recently used resources.

Flags: Read / Write

Default value: FALSE

Since: 2.10

The “sort-type” property

“sort-type” GtkRecentSortType

Sorting order to be used when displaying the recently used resources.

Flags: Read / Write

Default value: GTK_RECENT_SORT_NONE

Since: 2.10

Signal Details

The `"item-activated"` signal

```
void
user_function (GtkRecentChooser *chooser,
               gpointer          user_data)
```

This signal is emitted when the user "activates" a recent item in the recent chooser. This can happen by double-clicking on an item in the recently used resources list, or by pressing `Enter`.

Parameters

chooser	the object which received the signal
user_data	user data set when the signal handler was connected.

Flags: Run Last

Since: 2.10

The `"selection-changed"` signal

```
void
user_function (GtkRecentChooser *chooser,
               gpointer          user_data)
```

This signal is emitted when there is a change in the set of selected recently used resources. This can happen when a user modifies the selection with the mouse or the keyboard, or when explicitly calling functions to change the selection.

Parameters

chooser	the object which received the signal
user_data	user data set when the signal handler was connected.

Flags: Run Last

Since: 2.10

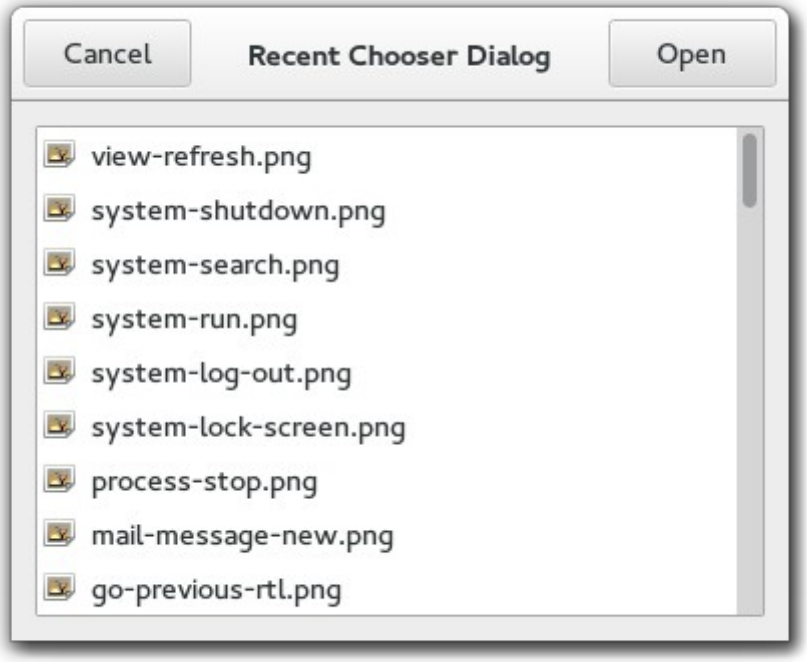
See Also

GtkRecentManager, GtkRecentChooserDialog, GtkRecentChooserWidget, GtkRecentChooserMenu

---

GtkRecentChooserDialog

GtkRecentChooserDialog — Displays recently used files in a dialog



Functions

GtkWidget *	gtk_recent_chooser_dialog_new ()
GtkWidget *	gtk_recent_chooser_dialog_new_for_manager ()

Types and Values

struct	GtkRecentChooserDialog

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkContainer
            └── GtkBin
                └── GtkWindow
                    └── GtkDialog
                        └── GtkRecentChooserDialog
```

Implemented Interfaces

GtkRecentChooserDialog implements AtkImplementorIface, GtkBuildable and GtkRecentChooser.

Includes

```
#include <gtk/gtk.h>
```

Description

GtkRecentChooserDialog is a dialog box suitable for displaying the recently used documents. This widgets works by putting a GtkRecentChooserWidget inside a GtkDialog. It exposes the GtkRecentChooserIface interface, so you can use all the GtkRecentChooser functions on the recent chooser dialog as well as those for GtkDialog.

Note that GtkRecentChooserDialog does not have any methods of its own. Instead, you should use the functions that work on a GtkRecentChooser.

Typical usage

In the simplest of cases, you can use the following code to use a GtkRecentChooserDialog to select a recently used file:

```
GtkWidget *dialog;
gint res;

dialog = gtk_recent_chooser_dialog_new
("Recent Documents",

parent_window,

_("_Cancel"),

GTK_RESPONSE_CANCEL,

_("_Open"),

GTK_RESPONSE_ACCEPT,

NULL);

res = gtk_dialog_run (GTK_DIALOG (dialog));
if (res == GTK_RESPONSE_ACCEPT)
  {
    GtkRecentInfo *info;
    GtkRecentChooser *chooser =
GTK_RECENT_CHOOSER (dialog);

    info =
gtk_recent_chooser_get_current_item
(chooser);
    open_file (gtk_recent_info_get_uri
(info));
    gtk_recent_info_unref (info);
  }

gtk_widget_destroy (dialog);
```

Recently used files are supported since GTK+ 2.10.

Functions

gtk_recent_chooser_dialog_new ()

```
GtkWidget *
```

```
gtk_recent_chooser_dialog_new (const gchar *title,
                               GtkWindow *parent,
                               const gchar *first_button_text,
                               ...);
```

Creates a new GtkRecentChooserDialog. This function is analogous to `gtk_dialog_new_with_buttons()`.

Parameters

title	Title of the dialog, or `NULL`.	[allow-none]
parent	Transient parent of the dialog, or `NULL`,.	[allow-none]
first_button_text	stock ID or text to go in the first button, or `NULL`.	[allow-none]
...	response ID for the first button, then additional (button, id) pairs, ending with `NULL`	

Returns

a new GtkRecentChooserDialog

Since: 2.10

---

gtk_recent_chooser_dialog_new_for_manager ()

```
GtkWidget *
gtk_recent_chooser_dialog_new_for_manager
                               (const gchar *title,
                                GtkWindow *parent,
                                GtkRecentManager *manager,
                                const gchar *first_button_text,
                                ...);
```

Creates a new GtkRecentChooserDialog with a specified recent manager.

This is useful if you have implemented your own recent manager, or if you have a customized instance of a GtkRecentManager object.

Parameters

title	Title of the dialog, or `NULL`.	[allow-none]
parent	Transient parent of the dialog, or `NULL`,.	[allow-none]
manager	a GtkRecentManager	
first_button_text	stock ID or text to go in the first button, or `NULL`.	[allow-none]
...	response ID for the first button, then additional (button, id) pairs, ending with `NULL`	

Returns

a new GtkRecentChooserDialog

Since: 2.10

Types and Values

struct GtkRecentChooserDialog

```
struct GtkRecentChooserDialog;
```

See Also

GtkRecentChooser, GtkDialog

GtkRecentChooserMenu

GtkRecentChooserMenu — Displays recently used files in a menu

Functions

GtkWidget *	gtk_recent_chooser_menu_new ()
GtkWidget *	gtk_recent_chooser_menu_new_for_manager ()
gboolean	gtk_recent_chooser_menu_get_show_numbers ()
void	gtk_recent_chooser_menu_set_show_numbers ()

Properties

gboolean	show-numbers	Read / Write

Types and Values

struct	GtkRecentChooserMenu

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkContainer
            └── GtkMenuShell
                └── GtkMenu
                    └── GtkRecentChooserMenu
```

Implemented Interfaces

GtkRecentChooserMenu implements AtkImplementorIface, GtkBuildable, GtkRecentChooser and GtkActivatable.

Includes

```
#include <gtk/gtk.h>
```

Description

GtkRecentChooserMenu is a widget suitable for displaying recently used files inside a menu. It can be used to set a sub-menu of a GtkMenuItem using `gtk_menu_item_set_submenu()`, or as the menu of a GtkMenuToolButton.

Note that GtkRecentChooserMenu does not have any methods of its own. Instead, you should use the functions that work on a GtkRecentChooser.

Note also that GtkRecentChooserMenu does not support multiple filters, as it has no way to let the user choose between them as the GtkRecentChooserWidget and GtkRecentChooserDialog widgets do. Thus using `gtk_recent_chooser_add_filter()` on a GtkRecentChooserMenu widget will yield the same effects as using `gtk_recent_chooser_set_filter()`, replacing any currently set filter with the supplied filter; `gtk_recent_chooser_remove_filter()` will remove any currently set GtkRecentFilter object and will unset the current filter; `gtk_recent_chooser_list_filters()` will return a list containing a single GtkRecentFilter object.

Recently used files are supported since GTK+ 2.10.

Functions

gtk_recent_chooser_menu_new ()

```
GtkWidget *
gtk_recent_chooser_menu_new (void);
```

Creates a new GtkRecentChooserMenu widget.

This kind of widget shows the list of recently used resources as a menu, each item as a menu item. Each item inside the menu might have an icon, representing its MIME type, and a number, for mnemonic access.

This widget implements the GtkRecentChooser interface.

This widget creates its own GtkRecentManager object. See the `gtk_recent_chooser_menu_new_for_manager()` function to know how to create a GtkRecentChooserMenu widget bound to another GtkRecentManager object.

Returns

a new GtkRecentChooserMenu

Since: 2.10

gtk_recent_chooser_menu_new_for_manager ()

```
GtkWidget *
gtk_recent_chooser_menu_new_for_manager
                                (GtkRecentManager *manager);
```

Creates a new GtkRecentChooserMenu widget using `manager` as the underlying recently used resources manager.

This is useful if you have implemented your own recent manager, or if you have a customized instance of a GtkRecentManager object or if you wish to share a common GtkRecentManager object among multiple GtkRecentChooser widgets.

Parameters

manager	a GtkRecentManager

Returns

a new GtkRecentChooserMenu, bound to `manager` .

Since: 2.10

---

gtk_recent_chooser_menu_get_show_numbers ()

```
gboolean
gtk_recent_chooser_menu_get_show_numbers
                                (GtkRecentChooserMenu *menu);
```

Returns the value set by `gtk_recent_chooser_menu_set_show_numbers()`.

Parameters

menu	a GtkRecentChooserMenu

Returns

`TRUE` if numbers should be shown.

Since: 2.10

---

gtk_recent_chooser_menu_set_show_numbers ()

```
void
gtk_recent_chooser_menu_set_show_numbers
                                (GtkRecentChooserMenu *menu,
                                 gboolean show_numbers);
```

Sets whether a number should be added to the items of `menu` . The numbers are shown to provide a unique character for a mnemonic to be used inside ten menu item’s label. Only the first the items get a number to avoid clashes.

Parameters

menu	a GtkRecentChooserMenu
show_numbers	whether to show numbers

Since: 2.10

Types and Values

struct GtkRecentChooserMenu

```
struct GtkRecentChooserMenu;
```

Property Details

The “show-numbers” property

“show-numbers”	gboolean

Whether the first ten items in the menu should be prepended by a number acting as a unique mnemonic.

Flags: Read / Write

Default value: FALSE

Since: 2.10

See Also

GtkRecentChooser

GtkRecentChooserWidget

GtkRecentChooserWidget — Displays recently used files

Functions

GtkWidget *	gtk_recent_chooser_widget_new ()
GtkWidget *	gtk_recent_chooser_widget_new_for_manager ()

Types and Values

struct	GtkRecentChooserWidget

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkContainer
            └── GtkBox
                └── GtkRecentChooserWidget
```

Implemented Interfaces

GtkRecentChooserWidget implements AtkImplementorIface, GtkBuildable, GtkOrientable and GtkRecentChooser.

Includes

```
#include <gtk/gtk.h>
```

Description

GtkRecentChooserWidget is a widget suitable for selecting recently used files. It is the main building block of a GtkRecentChooserDialog. Most applications will only need to use the latter; you can use GtkRecentChooserWidget as part of a larger window if you have special needs.

Note that GtkRecentChooserWidget does not have any methods of its own. Instead, you should use the functions that work on a GtkRecentChooser.

Recently used files are supported since GTK+ 2.10.

Functions

gtk_recent_chooser_widget_new ()

```
GtkWidget *
gtk_recent_chooser_widget_new (void);
```

Creates a new GtkRecentChooserWidget object. This is an embeddable widget used to access the recently used resources list.

Returns

a new GtkRecentChooserWidget

Since: 2.10

gtk_recent_chooser_widget_new_for_manager ()

```
GtkWidget *
gtk_recent_chooser_widget_new_for_manager
                                (GtkRecentManager *manager);
```

Creates a new GtkRecentChooserWidget with a specified recent manager.

This is useful if you have implemented your own recent manager, or if you have a customized instance of a GtkRecentManager object.

Parameters

manager	a GtkRecentManager

Returns

a new GtkRecentChooserWidget

Since: 2.10

Types and Values

struct GtkRecentChooserWidget

```
struct GtkRecentChooserWidget;
```

See Also

GtkRecentChooser, GtkRecentChooserDialog

GtkRecentFilter

GtkRecentFilter — A filter for selecting a subset of recently used files

Functions

gboolean	(*GtkRecentFilterFunc) ()
GtkRecentFilter *	gtk_recent_filter_new ()
const gchar *	gtk_recent_filter_get_name ()
void	gtk_recent_filter_set_name ()
void	gtk_recent_filter_add_mime_type ()
void	gtk_recent_filter_add_pattern ()
void	gtk_recent_filter_add_pixbuf_formats ()
void	gtk_recent_filter_add_application ()

void	gtk_recent_filter_add_group ()
void	gtk_recent_filter_add_age ()
void	gtk_recent_filter_add_custom ()
GtkRecentFilterFlags	gtk_recent_filter_get_needed ()
gboolean	gtk_recent_filter_filter ()

Types and Values

	GtkRecentFilter
struct	GtkRecentFilterInfo
enum	GtkRecentFilterFlags

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkRecentFilter
```

Implemented Interfaces

GtkRecentFilter implements GtkBuildable.

Includes

```
#include <gtk/gtk.h>
```

Description

A GtkRecentFilter can be used to restrict the files being shown in a GtkRecentChooser. Files can be filtered based on their name (with `gtk_recent_filter_add_pattern()`), on their mime type (with `gtk_file_filter_add_mime_type()`), on the application that has registered them (with `gtk_recent_filter_add_application()`), or by a custom filter function (with `gtk_recent_filter_add_custom()`).

Filtering by mime type handles aliasing and subclassing of mime types; e.g. a filter for text/plain also matches a file with mime type application/rtf, since application/rtf is a subclass of text/plain. Note that GtkRecentFilter allows wildcards for the subtype of a mime type, so you can e.g. filter for image/*.

Normally, filters are used by adding them to a GtkRecentChooser, see `gtk_recent_chooser_add_filter()`, but it is also possible to manually use a filter on a file with `gtk_recent_filter_filter()`.

Recently used files are supported since GTK+ 2.10.

GtkRecentFilter as GtkBuildable

The GtkRecentFilter implementation of the GtkBuildable interface supports adding rules using the <mime-types>, <patterns> and <applications> elements and listing the rules within. Specifying a <mime-type>, <pattern> or <application> has the same effect as calling `gtk_recent_filter_add_mime_type()`, `gtk_recent_filter_add_pattern()` or `gtk_recent_filter_add_application()`.

An example of a UI definition fragment specifying GtkRecentFilter rules:

```
<object class="GtkRecentFilter">
  <mime-types>
    <mime-type>text/plain</mime-type>
    <mime-type>image/png</mime-type>
  </mime-types>
  <patterns>
    <pattern>*.txt</pattern>
    <pattern>*.png</pattern>
  </patterns>
  <applications>
    <application>gimp</application>
    <application>gedit</application>
    <application>glade</application>
  </applications>
</object>
```

Functions

GtkRecentFilterFunc ()

```
gboolean
(*GtkRecentFilterFunc) (const GtkRecentFilterInfo *filter_info,
                        gpointer user_data);
```

The type of function that is used with custom filters, see `gtk_recent_filter_add_custom()`.

Parameters

filter_info	a GtkRecentFilterInfo that is filled according to the `needed` flags passed to `gtk_recent_filter_add_custom()`
user_data	user data passed to `gtk_recent_filter_add_custom()`

Returns

`TRUE` if the file should be displayed

---

gtk_recent_filter_new ()

```
GtkRecentFilter *
gtk_recent_filter_new (void);
```

Creates a new GtkRecentFilter with no rules added to it. Such filter does not accept any recently used resources, so is not particularly useful until you add rules with `gtk_recent_filter_add_pattern()`, `gtk_recent_filter_add_mime_type()`, `gtk_recent_filter_add_application()`, `gtk_recent_filter_add_age()`. To create a filter that accepts any recently used resource, use:

```
GtkRecentFilter *filter =
gtk_recent_filter_new ();
```

```
gtk_recent_filter_add_pattern (filter, "*");
```

Returns

a new GtkRecentFilter

Since: 2.10

---

gtk_recent_filter_get_name ()

```
const gchar *
gtk_recent_filter_get_name (GtkRecentFilter *filter);
```

Gets the human-readable name for the filter. See `gtk_recent_filter_set_name()`.

Parameters

filter	a GtkRecentFilter

Returns

the name of the filter, or `NULL`. The returned string is owned by the filter object and should not be freed.

[nullable]

Since: 2.10

---

gtk_recent_filter_set_name ()

```
void
gtk_recent_filter_set_name (GtkRecentFilter *filter,
                            const gchar *name);
```

Sets the human-readable name of the filter; this is the string that will be displayed in the recently used resources selector user interface if there is a selectable list of filters.

Parameters

filter	a GtkRecentFilter
name	then human readable name of `filter`

Since: 2.10

---

gtk_recent_filter_add_mime_type ()

```
void
gtk_recent_filter_add_mime_type (GtkRecentFilter *filter,
                                 const gchar *mime_type);
```

Adds a rule that allows resources based on their registered MIME type.

Parameters

filter	a GtkRecentFilter
mime_type	a MIME type

Since: 2.10

---

gtk_recent_filter_add_pattern ()

```
void
gtk_recent_filter_add_pattern (GtkRecentFilter *filter,
                               const gchar *pattern);
```

Adds a rule that allows resources based on a pattern matching their display name.

Parameters

filter	a GtkRecentFilter
pattern	a file pattern

Since: 2.10

---

gtk_recent_filter_add_pixbuf_formats ()

```
void
gtk_recent_filter_add_pixbuf_formats (GtkRecentFilter *filter);
```

Adds a rule allowing image files in the formats supported by GdkPixbuf.

Parameters

filter	a GtkRecentFilter

Since: 2.10

---

gtk_recent_filter_add_application ()

```
void
gtk_recent_filter_add_application (GtkRecentFilter *filter,
                                   const gchar *application);
```

Adds a rule that allows resources based on the name of the application that has registered them.

Parameters

filter	a GtkRecentFilter
application	an application name

Since: 2.10

---

gtk_recent_filter_add_group ()

```
void
gtk_recent_filter_add_group (GtkRecentFilter *filter,
                             const gchar *group);
```

Adds a rule that allows resources based on the name of the group to which they belong

Parameters

filter	a GtkRecentFilter
group	a group name

Since: 2.10

---

gtk_recent_filter_add_age ()

```
void
gtk_recent_filter_add_age (GtkRecentFilter *filter,
                           gint days);
```

Adds a rule that allows resources based on their age - that is, the number of days elapsed since they were last modified.

Parameters

filter	a GtkRecentFilter
days	number of days

Since: 2.10

---

gtk_recent_filter_add_custom ()

```
void
gtk_recent_filter_add_custom (GtkRecentFilter *filter,
                              GtkRecentFilterFlags needed,
                              GtkRecentFilterFunc func,
                              gpointer data,
                              GDestroyNotify data_destroy);
```

Adds a rule to a filter that allows resources based on a custom callback function. The bitfield `needed` which is passed in provides information about what sorts of information that the filter function needs; this allows GTK+ to avoid retrieving expensive information when it isn’t needed by the filter.

Parameters

filter	a GtkRecentFilter
needed	bitfield of flags indicating the

	information that the custom filter function needs.
func	callback function; if the function returns `TRUE`, then the file will be displayed.
data	data to pass to `func`
data_destroy	function to call to free `data` when it is no longer needed.

Since: 2.10

---

gtk_recent_filter_get_needed ()

```
GtkRecentFilterFlags
gtk_recent_filter_get_needed (GtkRecentFilter *filter);
```

Gets the fields that need to be filled in for the GtkRecentFilterInfo passed to `gtk_recent_filter_filter()`

This function will not typically be used by applications; it is intended principally for use in the implementation of GtkRecentChooser.

Parameters

filter	a GtkRecentFilter

Returns

bitfield of flags indicating needed fields when calling `gtk_recent_filter_filter()`

Since: 2.10

---

gtk_recent_filter_filter ()

```
gboolean
gtk_recent_filter_filter (GtkRecentFilter *filter,
                          const GtkRecentFilterInfo *filter_info);
```

Tests whether a file should be displayed according to `filter` . The GtkRecentFilterInfo `filter_info` should include the fields returned from `gtk_recent_filter_get_needed()`, and must set the GtkRecentFilterInfo.contains field of `filter_info` to indicate which fields have been set.

This function will not typically be used by applications; it is intended principally for use in the implementation of GtkRecentChooser.

Parameters

filter	a GtkRecentFilter
filter_info	a GtkRecentFilterInfo containing information about a recently used resource

Returns

`TRUE` if the file should be displayed

Since: 2.10

Types and Values

GtkRecentFilter

```
typedef struct _GtkRecentFilter GtkRecentFilter;
```

---

struct GtkRecentFilterInfo

```
struct GtkRecentFilterInfo {
  GtkRecentFilterFlags contains;

  const gchar *uri;
  const gchar *display_name;
  const gchar *mime_type;
  const gchar **applications;
  const gchar **groups;

  gint age;
};
```

A GtkRecentFilterInfo struct is used to pass information about the tested file to `gtk_recent_filter_filter()`.

Members

GtkRecentFilterFlags `contains`;	GtkRecentFilterFlags to indicate which fields are set.	
const gchar *`uri`;	The URI of the file being tested.	[nullable]
const gchar *`display_name`;	The string that will be used to display the file in the recent chooser.	[nullable]
const gchar *`mime_type`;	MIME type of the file.	[nullable]
const gchar **`applications`;	The list of applications that have registered the file.	[nullable][array zero-terminated=1]
const gchar **`groups`;	The groups to which the file belongs to.	[nullable][array zero-terminated=1]
gint `age`;	The number of days elapsed since the file has been registered.	

---

enum GtkRecentFilterFlags

These flags indicate what parts of a GtkRecentFilterInfo struct are filled or need to be filled.

Members

GTK_RECENT_FILTER_URI	the URI of the file being tested
GTK_RECENT_FILTER_DISPLAY_NAME	the string that will be used to display the file in the recent chooser
GTK_RECENT_FILTER_MIME_TYPE	the mime type of the file
GTK_RECENT_FILTER_APPLICATION	the list of applications that have registered the file
GTK_RECENT_FILTER_GROUP	the groups to which the file belongs to
GTK_RECENT_FILTER_AGE	the number of days elapsed since the file has been registered

---

Choosing from installed applications

GtkAppChooser — Interface implemented by widgets for choosing an application

GtkAppChooserButton — A button to launch an application chooser dialog

GtkAppChooserDialog — An application chooser dialog

GtkAppChooserWidget — Application chooser widget that can be embedded in other widgets

---

GtkAppChooser

GtkAppChooser — Interface implemented by widgets for choosing an application

Functions

GAppInfo *	gtk_app_chooser_get_app_info ()
gchar *	gtk_app_chooser_get_content_type ()
void	gtk_app_chooser_refresh ()

Properties

gchar *	content-type	Read / Write / Construct Only

Types and Values

GtkAppChooser

Object Hierarchy

```
GInterface
└── GtkAppChooser
```

Prerequisites

GtkAppChooser requires GtkWidget.

Known Implementations

GtkAppChooser is implemented by GtkAppChooserButton, GtkAppChooserDialog and GtkAppChooserWidget.

Includes

```
#include <gtk/gtk.h>
```

Description

GtkAppChooser is an interface that can be implemented by widgets which allow the user to choose an application (typically for the purpose of opening a file). The main objects that implement this interface are GtkAppChooserWidget, GtkAppChooserDialog and GtkAppChooserButton.

Applications are represented by GIO GAppInfo objects here. GIO has a concept of recommended and fallback applications for a given content type. Recommended applications are those that claim to handle the content type itself, while fallback also includes applications that handle a more generic content type. GIO also knows the default and last-used application for a given content type. The GtkAppChooserWidget provides detailed control over whether the shown list of applications should include default, recommended or fallback applications.

To obtain the application that has been selected in a GtkAppChooser, use `gtk_app_chooser_get_app_info()`.

Functions

gtk_app_chooser_get_app_info ()

```
GAppInfo *
gtk_app_chooser_get_app_info (GtkAppChooser *self);
```

Returns the currently selected application.

Parameters

self	a GtkAppChooser

Returns

a GAppInfo for the currently selected application, or `NULL` if none is selected. Free with `g_object_unref()`.

[nullable][transfer full]

Since: 3.0

---

gtk_app_chooser_get_content_type ()

```
gchar *
gtk_app_chooser_get_content_type (GtkAppChooser *self);
```

Returns the current value of the “content-type” property.

Parameters

self	a GtkAppChooser

Returns

the content type of `self` . Free with `g_free()`

Since: 3.0

---

gtk_app_chooser_refresh ()

```
void
gtk_app_chooser_refresh (GtkAppChooser *self);
```

Reloads the list of applications.

Parameters

self	a GtkAppChooser

Since: 3.0

Types and Values

GtkAppChooser

```
typedef struct _GtkAppChooser GtkAppChooser;
```

Property Details

The “content-type” property

“content-type”	gchar *

The content type of the GtkAppChooser object.

See GContentType for more information about content types.

Flags: Read / Write / Construct Only

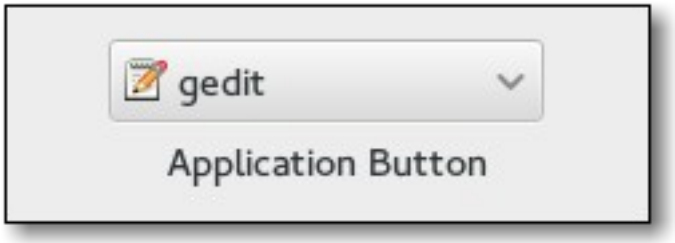
Default value: NULL

See Also

GAppInfo

GtkAppChooserButton

GtkAppChooserButton — A button to launch an application chooser dialog



Functions

GtkWidget *	gtk_app_chooser_button_new ()
void	gtk_app_chooser_button_append_custom_item ()
void	gtk_app_chooser_button_append_separator ()
void	gtk_app_chooser_button_set_active_custom_item ()
gboolean	gtk_app_chooser_button_get_show_default_item ()
void	gtk_app_chooser_button_set_show_default_item ()
gboolean	gtk_app_chooser_button_get_show_dialog_item ()
void	gtk_app_chooser_button_set_show_dialog_item ()
const gchar *	gtk_app_chooser_button_get_heading ()
void	gtk_app_chooser_button_set_heading ()

Properties

gchar *	heading	Read / Write
gboolean	show-default-item	Read / Write / Construct
gboolean	show-dialog-item	Read / Write / Construct

Signals

void	custom-item-activated	Has Details

Types and Values

struct	GtkAppChooserButton
struct	GtkAppChooserButtonClass

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkContainer
            └── GtkBin
                └── GtkComboBox
                    └── GtkAppChooserButton
```

Implemented Interfaces

GtkAppChooserButton implements AtkImplementorIface, GtkBuildable, GtkCellLayout, GtkCellEditable and GtkAppChooser.

Includes

```
#include <gtk/gtk.h>
```

Description

The GtkAppChooserButton is a widget that lets the user select an application. It implements the GtkAppChooser interface.

Initially, a GtkAppChooserButton selects the first application in its list, which will either be the most-recently used application or, if “show-default-item” is `TRUE`, the default application.

The list of applications shown in a GtkAppChooserButton includes the recommended applications for the given content type. When “show-default-item” is set, the default application is also included. To let the user chooser other applications, you can set the “show-dialog-item” property, which allows to open a full GtkAppChooserDialog.

It is possible to add custom items to the list, using `gtk_app_chooser_button_append_custom_item()`. These items cause the “custom-item-activated” signal to be emitted when they are selected.

To track changes in the selected application, use the “changed” signal.

Functions

gtk_app_chooser_button_new ()

```
GtkWidget *
gtk_app_chooser_button_new (const gchar *content_type);
```

Creates a new GtkAppChooserButton for applications that can handle content of the given type.

Parameters

content_type	the content type to show applications for

Returns

a newly created GtkAppChooserButton

Since: 3.0

---

gtk_app_chooser_button_append_custom_item ()

```
void
gtk_app_chooser_button_append_custom_item
                                (GtkAppChooserButton *self,
                                 const gchar *name,
                                 const gchar *label,
                                 GIcon *icon);
```

Appends a custom item to the list of applications that is shown in the popup; the item name must be unique per-widget. Clients can use the provided name as a detail for the "custom-item-activated" signal, to add a callback for the activation of a particular custom item in the list. See also `gtk_app_chooser_button_append_separator()`.

Parameters

self	a GtkAppChooserButton
name	the name of the custom item
label	the label for the custom item
icon	the icon for the custom item

Since: 3.0

---

gtk_app_chooser_button_append_separator ()

```
void
gtk_app_chooser_button_append_separator
                                (GtkAppChooserButton *self);
```

Appends a separator to the list of applications that is shown in the popup.

Parameters

self	a GtkAppChooserButton

Since: 3.0

---

gtk_app_chooser_button_set_active_custom_item ()

```
void
gtk_app_chooser_button_set_active_custom_item
                                (GtkAppChooserButton *self,
                                 const gchar *name);
```

Selects a custom item previously added with `gtk_app_chooser_button_append_custom_item()`.

Use `gtk_app_chooser_refresh()` to bring the selection to its initial state.

Parameters

self	a GtkAppChooserButton
name	the name of the custom item

Since: 3.0

---

gtk_app_chooser_button_get_show_default_item ()

```
gboolean
gtk_app_chooser_button_get_show_default_item
                                (GtkAppChooserButton *self);
```

Returns the current value of the “show-default-item” property.

Parameters

self	a GtkAppChooserButton

Returns

the value of “show-default-item”

Since: 3.2

---

gtk_app_chooser_button_set_show_default_item ()

```
void
gtk_app_chooser_button_set_show_default_item
                               (GtkAppChooserButton *self,
                                gboolean setting);
```

Sets whether the dropdown menu of this button should show the default application for the given content type at top.

Parameters

self	a GtkAppChooserButton
setting	the new value for “show-default-item”

Since: 3.2

---

gtk_app_chooser_button_get_show_dialog_item ()

```
gboolean
```

```
gtk_app_chooser_button_get_show_dialog_item
                                (GtkAppChooserButton *self);
```

Returns the current value of the “show-dialog-item” property.

Parameters

self	a GtkAppChooserButton

Returns

the value of “show-dialog-item”

Since: 3.0

---

gtk_app_chooser_button_set_show_dialog_item ()

```
void
gtk_app_chooser_button_set_show_dialog_item
                                (GtkAppChooserButton *self,
                                 gboolean setting);
```

Sets whether the dropdown menu of this button should show an entry to trigger a GtkAppChooserDialog.

Parameters

self	a GtkAppChooserButton
setting	the new value for “show-dialog-item”

Since: 3.0

---

gtk_app_chooser_button_get_heading ()

```
const gchar *
gtk_app_chooser_button_get_heading (GtkAppChooserButton *self);
```

Returns the text to display at the top of the dialog.

Parameters

self	a GtkAppChooserButton

Returns

the text to display at the top of the dialog, or `NULL`, in which case a default text is displayed.

[nullable]

---

gtk_app_chooser_button_set_heading ()

```
void
gtk_app_chooser_button_set_heading (GtkAppChooserButton *self,
                                    const gchar *heading);
```

Sets the text to display at the top of the dialog. If the heading is not set, the dialog displays a default text.

Parameters

self	a GtkAppChooserButton
heading	a string containing Pango markup

Types and Values

struct GtkAppChooserButton

```
struct GtkAppChooserButton;
```

---

struct GtkAppChooserButtonClass

```
struct GtkAppChooserButtonClass {
  GtkComboBoxClass parent_class;


  void (* custom_item_activated) (GtkAppChooserButton *self,
                                  const gchar *item_name);
};
```

Members

`custom_item_activated` ()	Signal emitted when a custom item, previously added with `gtk_app_chooser_button_append_custom_item()`, is activated from the dropdown menu.

Property Details

The `“heading”` property

```
  “heading”                  gchar *
```

The text to show at the top of the dialog that can be opened from the button. The string may contain Pango markup.

Flags: Read / Write

Default value: NULL

The “show-default-item” property

```
  “show-default-item”        gboolean
```

The “show-default-item” property determines whether the dropdown menu should show the default application on top for the provided content type.

Flags: Read / Write / Construct

Default value: FALSE

Since: 3.2

The “show-dialog-item” property

```
  “show-dialog-item”         gboolean
```

The “show-dialog-item” property determines whether the dropdown menu should show an item that triggers a GtkAppChooserDialog when clicked.

Flags: Read / Write / Construct

Default value: FALSE

Signal Details

The “custom-item-activated” signal

```
void
user_function (GtkAppChooserButton *self,
               gchar               *item_name,
               gpointer             user_data)
```

Emitted when a custom item, previously added with `gtk_app_chooser_button_append_custom_item()`, is activated from the dropdown menu.

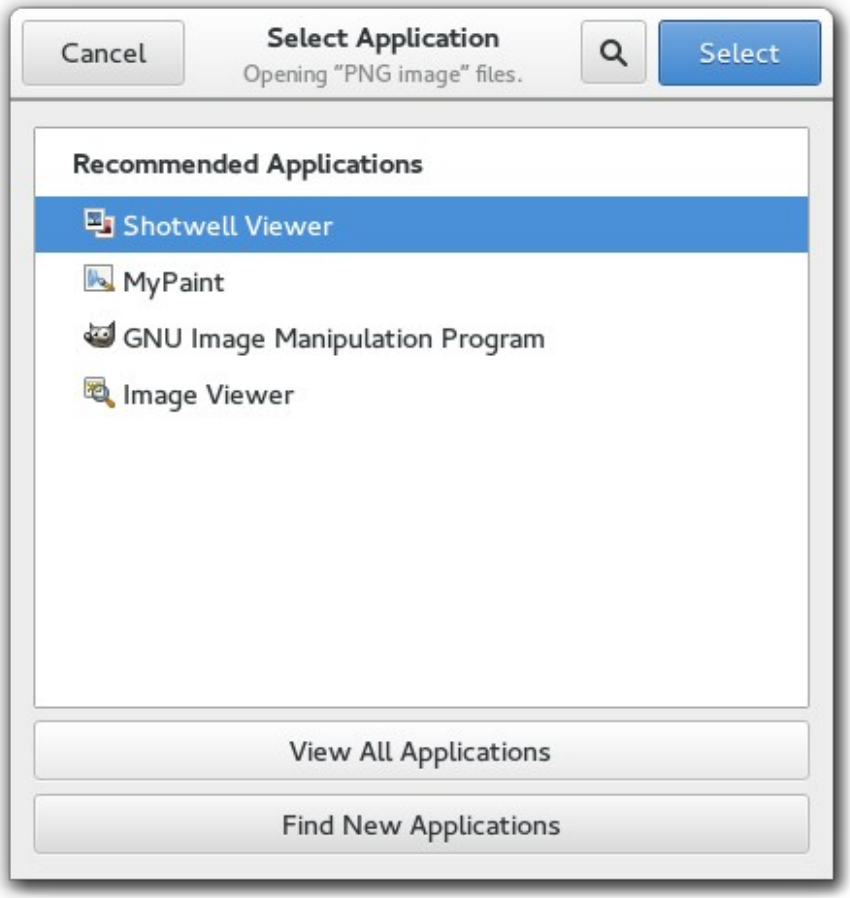
Parameters

self	the object which received the signal
item_name	the name of the activated item
user_data	user data set when the signal handler was connected.

Flags: Has Details

GtkAppChooserDialog

GtkAppChooserDialog — An application chooser dialog



Functions

GtkWidget *	gtk_app_chooser_dialog_new ()
GtkWidget *	gtk_app_chooser_dialog_new_for_content_type ()
GtkWidget *	gtk_app_chooser_dialog_get_widget ()
void	gtk_app_chooser_dialog_set_heading ()
const gchar *	gtk_app_chooser_dialog_get_heading ()

Properties

GFile *	gfile	Read / Write / Construct Only
gchar *	heading	Read / Write

Types and Values

struct	GtkAppChooserDialog
struct	GtkAppChooserDialogClass

Object Hierarchy

```
GObject
└── GInitiallyUnowned
```

```
        └── GtkWidget
            └── GtkContainer
                └── GtkBin
                    └── GtkWindow
                        └── GtkDialog
                            └── GtkAppChooserDialog
```

Implemented Interfaces

GtkAppChooserDialog implements AtkImplementorIface, GtkBuildable and GtkAppChooser.

Includes

```
#include <gtk/gtk.h>
```

Description

GtkAppChooserDialog shows a GtkAppChooserWidget inside a GtkDialog.

Note that GtkAppChooserDialog does not have any interesting methods of its own. Instead, you should get the embedded GtkAppChooserWidget using `gtk_app_chooser_dialog_get_widget()` and call its methods if the generic GtkAppChooser interface is not sufficient for your needs.

To set the heading that is shown above the GtkAppChooserWidget, use `gtk_app_chooser_dialog_set_heading()`.

Functions

gtk_app_chooser_dialog_new ()

```
GtkWidget *
gtk_app_chooser_dialog_new (GtkWindow *parent,
                            GtkDialogFlags flags,
                            GFile *file);
```

Creates a new GtkAppChooserDialog for the provided GFile, to allow the user to select an application for it.

Parameters

parent	a GtkWindow, or `NULL`.	[allow-none]
flags	flags for this dialog	
file	a GFile	

Returns

a newly created GtkAppChooserDialog

Since: 3.0

gtk_app_chooser_dialog_new_for_content_type ()

```
GtkWidget *
gtk_app_chooser_dialog_new_for_content_type
                                (GtkWindow *parent,
                                 GtkDialogFlags flags,
                                 const gchar *content_type);
```

Creates a new GtkAppChooserDialog for the provided content type, to allow the user to select an application for it.

Parameters

parent	a GtkWindow, or `NULL`.	[allow-none]
flags	flags for this dialog	
content_type	a content type string	

Returns

a newly created GtkAppChooserDialog

Since: 3.0

---

gtk_app_chooser_dialog_get_widget ()

```
GtkWidget *
gtk_app_chooser_dialog_get_widget (GtkAppChooserDialog *self);
```

Returns the GtkAppChooserWidget of this dialog.

Parameters

self	a GtkAppChooserDialog

Returns

the GtkAppChooserWidget of `self` .

[transfer none]

Since: 3.0

---

gtk_app_chooser_dialog_set_heading ()

```
void
gtk_app_chooser_dialog_set_heading (GtkAppChooserDialog *self,
                                    const gchar *heading);
```

Sets the text to display at the top of the dialog. If the heading is not set, the dialog displays a default text.

Parameters

self	a GtkAppChooserDialog
heading	a string containing Pango markup

---

gtk_app_chooser_dialog_get_heading ()

```
const gchar *
gtk_app_chooser_dialog_get_heading (GtkAppChooserDialog *self);
```

Returns the text to display at the top of the dialog.

Parameters

self	a GtkAppChooserDialog

Returns

the text to display at the top of the dialog, or `NULL`, in which case a default text is displayed.

[nullable]

Types and Values

struct GtkAppChooserDialog

```
struct GtkAppChooserDialog;
```

---

struct GtkAppChooserDialogClass

```
struct GtkAppChooserDialogClass {
  GtkDialogClass parent_class;
};
```

Members

Property Details

The “gfile” property

```
  “gfile”                    GFile *
```

The GFile used by the GtkAppChooserDialog. The dialog's GtkAppChooserWidget content type will be guessed from the file, if present.

Flags: Read / Write / Construct Only

The “heading” property

“heading” gchar *

The text to show at the top of the dialog. The string may contain Pango markup.

Flags: Read / Write

Default value: NULL

GtkAppChooserWidget

GtkAppChooserWidget — Application chooser widget that can be embedded in other widgets

Functions

GtkWidget *	gtk_app_chooser_widget_new ()
void	gtk_app_chooser_widget_set_show_default ()
gboolean	gtk_app_chooser_widget_get_show_default ()
void	gtk_app_chooser_widget_set_show_recommended ()
gboolean	gtk_app_chooser_widget_get_show_recommended ()
void	gtk_app_chooser_widget_set_show_fallback ()
gboolean	gtk_app_chooser_widget_get_show_fallback ()
void	gtk_app_chooser_widget_set_show_other ()
gboolean	gtk_app_chooser_widget_get_show_other ()
void	gtk_app_chooser_widget_set_show_all ()
gboolean	gtk_app_chooser_widget_get_show_all ()
void	gtk_app_chooser_widget_set_default_text ()
const gchar *	gtk_app_chooser_widget_get_default_text ()

Properties

gchar *	default-text	Read / Write
gboolean	show-all	Read / Write / Construct
gboolean	show-default	Read / Write / Construct
gboolean	show-fallback	Read / Write / Construct
gboolean	show-other	Read / Write / Construct
gboolean	show-recommended	Read / Write / Construct

Signals

void	application-activated	Run First

void	application-selected	Run First
void	populate-popup	Run First

Types and Values

struct	GtkAppChooserWidget
struct	GtkAppChooserWidgetClass

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkContainer
            └── GtkBox
                └── GtkAppChooserWidget
```

Implemented Interfaces

GtkAppChooserWidget implements AtkImplementorIface, GtkBuildable, GtkOrientable and GtkAppChooser.

Includes

```
#include <gtk/gtk.h>
```

Description

GtkAppChooserWidget is a widget for selecting applications. It is the main building block for GtkAppChooserDialog. Most applications only need to use the latter; but you can use this widget as part of a larger widget if you have special needs.

GtkAppChooserWidget offers detailed control over what applications are shown, using the “show-default”, “show-recommended”, “show-fallback”, “show-other” and “show-all” properties. See the GtkAppChooser documentation for more information about these groups of applications.

To keep track of the selected application, use the “application-selected” and “application-activated” signals.

CSS nodes

GtkAppChooserWidget has a single CSS node with name appchooser.

Functions

gtk_app_chooser_widget_new ()

```
GtkWidget *
gtk_app_chooser_widget_new (const gchar *content_type);
```

Creates a new GtkAppChooserWidget for applications that can handle content of the given type.

Parameters

content_type	the content type to show applications for

Returns

a newly created GtkAppChooserWidget

Since: 3.0

---

gtk_app_chooser_widget_set_show_default ()

```
void
gtk_app_chooser_widget_set_show_default
                                (GtkAppChooserWidget *self,
                                 gboolean setting);
```

Sets whether the app chooser should show the default handler for the content type in a separate section.

Parameters

self	a GtkAppChooserWidget
setting	the new value for “show-default”

Since: 3.0

---

gtk_app_chooser_widget_get_show_default ()

```
gboolean
gtk_app_chooser_widget_get_show_default
                                (GtkAppChooserWidget *self);
```

Returns the current value of the “show-default” property.

Parameters

self	a GtkAppChooserWidget

Returns

the value of “show-default”

Since: 3.0

---

gtk_app_chooser_widget_set_show_recommended ()

```
void
gtk_app_chooser_widget_set_show_recommended
                                (GtkAppChooserWidget *self,
                                 gboolean setting);
```

Sets whether the app chooser should show recommended applications for the content type in a separate section.

Parameters

self	a GtkAppChooserWidget
setting	the new value for “show-recommended”

Since: 3.0

---

gtk_app_chooser_widget_get_show_recommended ()

```
gboolean
gtk_app_chooser_widget_get_show_recommended
                                (GtkAppChooserWidget *self);
```

Returns the current value of the “show-recommended” property.

Parameters

self	a GtkAppChooserWidget

Returns

the value of “show-recommended”

Since: 3.0

---

gtk_app_chooser_widget_set_show_fallback ()

```
void
gtk_app_chooser_widget_set_show_fallback
                                (GtkAppChooserWidget *self,
                                 gboolean setting);
```

Sets whether the app chooser should show related applications for the content type in a separate section.

Parameters

self	a GtkAppChooserWidget
setting	the new value for “show-fallback”

Since: 3.0

---

gtk_app_chooser_widget_get_show_fallback ()

```
gboolean
gtk_app_chooser_widget_get_show_fallback
                                (GtkAppChooserWidget *self);
```

Returns the current value of the “show-fallback” property.

Parameters

self	a GtkAppChooserWidget

Returns

the value of “show-fallback”

Since: 3.0

---

gtk_app_chooser_widget_set_show_other ()

```
void
gtk_app_chooser_widget_set_show_other (GtkAppChooserWidget *self,
                                       gboolean setting);
```

Sets whether the app chooser should show applications which are unrelated to the content type.

Parameters

self	a GtkAppChooserWidget
setting	the new value for “show-other”

Since: 3.0

---

gtk_app_chooser_widget_get_show_other ()

```
gboolean
gtk_app_chooser_widget_get_show_other (GtkAppChooserWidget *self);
```

Returns the current value of the “show-other” property.

Parameters

self	a GtkAppChooserWidget

Returns

the value of “show-other”

Since: 3.0

---

gtk_app_chooser_widget_set_show_all ()

```
void
gtk_app_chooser_widget_set_show_all (GtkAppChooserWidget *self,
                                     gboolean setting);
```

Sets whether the app chooser should show all applications in a flat list.

Parameters

self	a GtkAppChooserWidget
setting	the new value for “show-all”

Since: 3.0

---

gtk_app_chooser_widget_get_show_all ()

```
gboolean
gtk_app_chooser_widget_get_show_all (GtkAppChooserWidget *self);
```

Returns the current value of the “show-all” property.

Parameters

self	a GtkAppChooserWidget

Returns

the value of “show-all”

Since: 3.0

---

gtk_app_chooser_widget_set_default_text ()

```
void
gtk_app_chooser_widget_set_default_text
                                (GtkAppChooserWidget *self,
                                 const gchar *text);
```

Sets the text that is shown if there are not applications that can handle the content type.

Parameters

self	a GtkAppChooserWidget
text	the new value for “default-text”

---

gtk_app_chooser_widget_get_default_text ()

```
const gchar *
gtk_app_chooser_widget_get_default_text
                                (GtkAppChooserWidget *self);
```

Returns the text that is shown if there are not applications that can handle the content type.

Parameters

self	a GtkAppChooserWidget

Returns

the value of "default-text"

Since: 3.0

Types and Values

struct GtkAppChooserWidget

```
struct GtkAppChooserWidget;
```

---

struct GtkAppChooserWidgetClass

```
struct GtkAppChooserWidgetClass {
  GtkBoxClass parent_class;


  void (* application_selected)  (GtkAppChooserWidget *self,
                                  GAppInfo            *app_info);

  void (* application_activated) (GtkAppChooserWidget *self,
                                  GAppInfo            *app_info);

  void (* populate_popup)        (GtkAppChooserWidget *self,
                                  GtkMenu             *menu,
                                  GAppInfo            *app_info);
};
```

Members

application_selected ()	Signal emitted when an application item is selected from the widget's list.
application_activated ()	Signal emitted when an application item is activated from the widget's list.
populate_popup ()	Signal emitted when a context menu

	is about to popup over an application item.

Property Details

The “default-text” property

“default-text”	gchar *

The “default-text” property determines the text that appears in the widget when there are no applications for the given content type. See also gtk_app_chooser_widget_set_default_text().

Flags: Read / Write

Default value: NULL

The “show-all” property

“show-all”	gboolean

If the “show-all” property is TRUE, the app chooser presents all applications in a single list, without subsections for default, recommended or related applications.

Flags: Read / Write / Construct

Default value: FALSE

The “show-default” property

“show-default”	gboolean

The ::show-default property determines whether the app chooser should show the default handler for the content type in a separate section. If FALSE, the default handler is listed among the recommended applications.

Flags: Read / Write / Construct

Default value: FALSE

The “show-fallback” property

“show-fallback”	gboolean

The “show-fallback” property determines whether the app chooser should show a section for fallback applications. If FALSE, the fallback applications are listed among the other applications.

Flags: Read / Write / Construct

Default value: FALSE

The “show-other” property

```
  “show-other”               gboolean
```

The “show-other” property determines whether the app chooser should show a section for other applications.

Flags: Read / Write / Construct

Default value: FALSE

The “show-recommended” property

```
  “show-recommended”         gboolean
```

The “show-recommended” property determines whether the app chooser should show a section for recommended applications. If `FALSE`, the recommended applications are listed among the other applications.

Flags: Read / Write / Construct

Default value: TRUE

Signal Details

The “application-activated” signal

```
void
user_function (GtkAppChooserWidget *self,
               GAppInfo            *application,
               gpointer             user_data)
```

Emitted when an application item is activated from the widget's list.

This usually happens when the user double clicks an item, or an item is selected and the user presses one of the keys Space, Shift+Space, Return or Enter.

Parameters

self	the object which received the signal
application	the activated GAppInfo
user_data	user data set when the signal handler was connected.

Flags: Run First

The “application-selected” signal

```
void
user_function (GtkAppChooserWidget *self,
               GAppInfo            *application,
               gpointer             user_data)
```

Emitted when an application item is selected from the widget's list.

Parameters

self	the object which received the signal
application	the selected GAppInfo
user_data	user data set when the signal handler was connected.

Flags: Run First

The `"populate-popup"` signal

```
void
user_function (GtkAppChooserWidget *self,
               GtkMenu             *menu,
               GAppInfo            *application,
               gpointer             user_data)
```

Emitted when a context menu is about to popup over an application item. Clients can insert menu items into the provided GtkMenu object in the callback of this signal; the context menu will be shown over the item if at least one item has been added to the menu.

Parameters

self	the object which received the signal
menu	the GtkMenu to populate
application	the current GAppInfo
user_data	user data set when the signal handler was connected.

Flags: Run First

Gestures and event handling

GtkEventController — Self-contained handler of series of events

GtkEventControllerKey — Event controller for key events

GtkEventControllerScroll — Event controller for scroll events

GtkEventControllerMotion — Event controller for motion events

GtkGesture — Base class for gestures

GtkGestureSingle — Base class for mouse/single-touch gestures

GtkGestureDrag — Drag gesture

GtkGestureLongPress — "Press and Hold" gesture

GtkGestureMultiPress — Multipress gesture

GtkGesturePan — Pan gesture

GtkGestureSwipe — Swipe gesture

GtkGestureRotate — Rotate gesture

GtkGestureZoom — Zoom gesture

GtkGestureStylus — Gesture for stylus input

GtkPadController — Controller for drawing tablet pads

GtkEventController

GtkEventController — Self-contained handler of series of events

Functions

GtkPropagationPhase	gtk_event_controller_get_propagation_phase ()
void	gtk_event_controller_set_propagation_phase ()
gboolean	gtk_event_controller_handle_event ()
GtkWidget *	gtk_event_controller_get_widget ()
void	gtk_event_controller_reset ()

Properties

GtkPropagationPhase	propagation-phase	Read / Write
GtkWidget *	widget	Read / Write / Construct Only

Types and Values

	GtkEventController
enum	GtkPropagationPhase

Object Hierarchy

```
GObject
└── GtkEventController
    ├── GtkEventControllerKey
    ├── GtkEventControllerMotion
    ├── GtkEventControllerScroll
    ├── GtkGesture
    └── GtkPadController
```

Includes

```
#include <gtk/gtk.h>
```

Description

GtkEventController is a base, low-level implementation for event controllers. Those react to a series of GdkEvents, and possibly trigger actions as a consequence of those.

Functions

gtk_event_controller_get_propagation_phase ()

```
GtkPropagationPhase
gtk_event_controller_get_propagation_phase
                                (GtkEventController *controller);
```

Gets the propagation phase at which `controller` handles events.

Parameters

controller	a GtkEventController

Returns

the propagation phase

Since: 3.14

---

gtk_event_controller_set_propagation_phase ()

```
void
gtk_event_controller_set_propagation_phase
                                (GtkEventController *controller,
                                 GtkPropagationPhase phase);
```

Sets the propagation phase at which a controller handles events.

If `phase` is `GTK_PHASE_NONE`, no automatic event handling will be performed, but other additional gesture maintenance will. In that phase, the events can be managed by calling `gtk_event_controller_handle_event()`.

Parameters

controller	a GtkEventController
phase	a propagation phase

Since: 3.14

---

gtk_event_controller_handle_event ()

```
gboolean
gtk_event_controller_handle_event (GtkEventController *controller,
                                   const GdkEvent *event);
```

Feeds an events into `controller` , so it can be interpreted and the controller actions triggered.

Parameters

controller	a GtkEventController
event	a GdkEvent

Returns

`TRUE` if the event was potentially useful to trigger the controller action

Since: 3.14

---

gtk_event_controller_get_widget ()

```
GtkWidget *
gtk_event_controller_get_widget (GtkEventController *controller);
```

Returns the GtkWidget this controller relates to.

Parameters

controller	a GtkEventController

Returns

a GtkWidget.

[transfer none]

Since: 3.14

---

gtk_event_controller_reset ()

```
void
gtk_event_controller_reset (GtkEventController *controller);
```

Resets the `controller` to a clean state. Every interaction the controller did through “handle-event” will be dropped at this point.

Parameters

controller	a GtkEventController

Since: 3.14

Types and Values

GtkEventController

```
typedef struct _GtkEventController GtkEventController;
```

---

enum GtkPropagationPhase

Describes the stage at which events are fed into a GtkEventController.

Members

GTK_PHASE_NONE	Events are not delivered automatically. Those can be manually fed through `gtk_event_controller_handle_event()`. This should only be used when full control about when, or whether the controller handles the event is needed.
GTK_PHASE_CAPTURE	Events are delivered in the capture phase. The capture phase happens before the bubble phase, runs from the toplevel down to the event widget. This option should only be used on containers that might possibly handle events before their children do.
GTK_PHASE_BUBBLE	Events are delivered in the bubble phase. The bubble phase happens after the capture phase, and before the default handlers are run. This phase runs from the event widget, up to the toplevel.
GTK_PHASE_TARGET	Events are delivered in the default widget event handlers, note that widget implementations must chain up on button, motion, touch and grab broken handlers for controllers in this phase to be run.

Since: 3.14

Property Details

The `“propagation-phase”` property

`“propagation-phase” GtkPropagationPhase`

The propagation phase at which this controller will handle events.

Flags: Read / Write

Default value: GTK_PHASE_BUBBLE

Since: 3.14

The “widget” property

```
  “widget”                   GtkWidget *
```

The widget receiving the GdkEvents that the controller will handle.

Flags: Read / Write / Construct Only

Since: 3.14

See Also

GtkGesture

GtkEventControllerKey

GtkEventControllerKey — Event controller for key events

Functions

GtkEventController *	gtk_event_controller_key_new ()

Signals

void	focus-in	Run Last
void	focus-out	Run Last
void	im-update	Run Last
gboolean	key-pressed	Run Last
void	key-released	Run Last
gboolean	modifiers	Run Last

Types and Values

GtkEventControllerKey

Object Hierarchy

```
    GObject
    ╰── GtkEventController
        ╰── GtkEventControllerKey
```

Includes

```
#include <gtk/gtk.h>
```

Description

GtkEventControllerKey is an event controller meant for situations where you need access to key events.

This object was added in 3.24.

Functions

gtk_event_controller_key_new ()

```
GtkEventController *
gtk_event_controller_key_new (GtkWidget *widget);
```

Types and Values

GtkEventControllerKey

```
typedef struct _GtkEventControllerKey GtkEventControllerKey;
```

Signal Details

The “focus-in” signal

```
void
user_function (GtkEventControllerKey *eventcontrollerkey,
               gpointer               user_data)
```

Flags: Run Last

The “focus-out” signal

```
void
user_function (GtkEventControllerKey *eventcontrollerkey,
               gpointer               user_data)
```

Flags: Run Last

The “im-update” signal

```
void
user_function (GtkEventControllerKey *eventcontrollerkey,
               gpointer               user_data)
```

Flags: Run Last

The “key-pressed” signal

```
gboolean
user_function (GtkEventControllerKey *controller,
               guint                  keyval,
               guint                  keycode,
               GdkModifierType        state,
               gpointer               user_data)
```

This signal is emitted whenever a key is pressed.

Parameters

controller	the object which received the signal.
keyval	the pressed key.
keycode	the raw code of the pressed key.
state	the bitmask, representing the state of modifier keys and pointer buttons. See GdkModifierType.
user_data	user data set when the signal handler was connected.

Returns

`TRUE` if the key press was handled, `FALSE` otherwise.

Flags: Run Last

Since: 3.24

The “key-released” signal

```
void
user_function (GtkEventControllerKey *controller,
               guint                  keyval,
               guint                  keycode,
               GdkModifierType        state,
               gpointer               user_data)
```

This signal is emitted whenever a key is released.

Parameters

controller	the object which received the signal.
keyval	the released key.
keycode	the raw code of the released key.
state	the bitmask, representing the state of modifier keys and pointer

	buttons. See GdkModifierType.
user_data	user data set when the signal handler was connected.

Flags: Run Last

Since: 3.24

The “modifiers” signal

```
gboolean
user_function (GtkEventControllerKey *eventcontrollerkey,
               GdkModifierType        arg1,
               gpointer               user_data)
```

Flags: Run Last

See Also

GtkEventController

GtkEventControllerScroll

GtkEventControllerScroll — Event controller for scroll events

Functions

GtkEventController *	gtk_event_controller_scroll_new ()
void	gtk_event_controller_scroll_set_flags ()
GtkEventControllerScrollFlags	gtk_event_controller_scroll_get_flags ()

Properties

GtkEventControllerScrollFlags	flags	Read / Write

Signals

void	decelerate	Run First
void	scroll	Run First
void	scroll-begin	Run First
void	scroll-end	Run First

Types and Values

	GtkEventControllerScroll
enum	GtkEventControllerScrollFlags

Object Hierarchy

```
GObject
└── GtkEventController
    └── GtkEventControllerScroll
```

Includes

```
#include <gtk/gtk.h>
```

Description

GtkEventControllerScroll is an event controller meant to handle scroll events from mice and touchpads. It is capable of handling both discrete and continuous scroll events, abstracting them both on the “scroll” signal (deltas in the discrete case are multiples of 1).

In the case of continuous scroll events, GtkEventControllerScroll encloses all “scroll” events between two “scroll-begin” and “scroll-end” signals.

The behavior of the event controller can be modified by the flags given at creation time, or modified at a later point through `gtk_event_controller_scroll_set_flags()` (e.g. because the scrolling conditions of the widget changed).

The controller can be set up to emit motion for either/both vertical and horizontal scroll events through GTK_EVENT_CONTROLLER_SCROLL_VERTICAL, GTK_EVENT_CONTROLLER_SCROLL_HORIZONTAL and GTK_EVENT_CONTROLLER_SCROLL_BOTH. If any axis is disabled, the respective “scroll” delta will be 0. Vertical scroll events will be translated to horizontal motion for the devices incapable of horizontal scrolling.

The event controller can also be forced to emit discrete events on all devices through GTK_EVENT_CONTROLLER_SCROLL_DISCRETE. This can be used to implement discrete actions triggered through scroll events (e.g. switching across combobox options).

The GTK_EVENT_CONTROLLER_SCROLL_KINETIC flag toggles the emission of the “decelerate” signal, emitted at the end of scrolling with two X/Y velocity arguments that are consistent with the motion that was received.

This object was added in 3.24.

Functions

gtk_event_controller_scroll_new ()

```
GtkEventController *
gtk_event_controller_scroll_new (GtkWidget *widget,
                                 GtkEventControllerScrollFlags flags);
```

Creates a new event controller that will handle scroll events for the given `widget` .

Parameters

widget	a GtkWidget

flags	behavior flags

Returns

a new GtkEventControllerScroll

Since: 3.24

---

gtk_event_controller_scroll_set_flags ()

```
void
gtk_event_controller_scroll_set_flags (GtkEventControllerScroll *controller,
                                       GtkEventControllerScrollFlags flags);
```

Sets the flags conditioning scroll controller behavior.

Parameters

scroll	a GtkEventControllerScroll
flags	behavior flags

Since: 3.24

---

gtk_event_controller_scroll_get_flags ()

```
GtkEventControllerScrollFlags
gtk_event_controller_scroll_get_flags (GtkEventControllerScroll *controller);
```

Gets the flags conditioning the scroll controller behavior.

Parameters

scroll	a GtkEventControllerScroll

Returns

the controller flags.

Since: 3.24

Types and Values

GtkEventControllerScroll

```
typedef struct _GtkEventControllerScroll GtkEventControllerScroll;
```

---

enum GtkEventControllerScrollFlags

Describes the behavior of a GtkEventControllerScroll.

Members

GTK_EVENT_CONTROLLER_SCROLL_NONE	Don't emit scroll.
GTK_EVENT_CONTROLLER_SCROLL_VERTICAL	Emit scroll with vertical deltas.
GTK_EVENT_CONTROLLER_SCROLL_HORIZONTAL	Emit scroll with horizontal deltas.
GTK_EVENT_CONTROLLER_SCROLL_DISCRETE	Only emit deltas that are multiples of 1.
GTK_EVENT_CONTROLLER_SCROLL_KINETIC	Emit “decelerate” after continuous scroll finishes.
GTK_EVENT_CONTROLLER_SCROLL_BOTH_AXES	Emit scroll on both axes.

Since: 3.24

Property Details

The `“flags”` property

```
  “flags”                    GtkEventControllerScrollFlags
```

The flags affecting event controller behavior

Flags: Read / Write

Since: 3.24

Signal Details

The `“decelerate”` signal

```
void
user_function (GtkEventControllerScroll *controller,
               gdouble                   vel_x,
               gdouble                   vel_y,
               gpointer                  user_data)
```

Emitted after scroll is finished if the GTK_EVENT_CONTROLLER_SCROLL_KINETIC flag is set. `vel_x` and `vel_y` express the initial velocity that was imprinted by the scroll events. `vel_x` and `vel_y` are expressed in pixels/ms.

Parameters

controller	The object that received the signal

vel_x	X velocity
vel_y	Y velocity
user_data	user data set when the signal handler was connected.

Flags: Run First

The `"scroll"` signal

```
void
user_function (GtkEventControllerScroll *controller,
               gdouble                   dx,
               gdouble                   dy,
               gpointer                  user_data)
```

Signals that the widget should scroll by the amount specified by `dx` and `dy` .

Parameters

controller	The object that received the signal
dx	X delta
dy	Y delta
user_data	user data set when the signal handler was connected.

Flags: Run First

The `"scroll-begin"` signal

```
void
user_function (GtkEventControllerScroll *controller,
               gpointer                  user_data)
```

Signals that a new scrolling operation has begun. It will only be emitted on devices capable of it.

Parameters

controller	The object that received the signal
user_data	user data set when the signal handler was connected.

Flags: Run First

The `"scroll-end"` signal

```
void
user_function (GtkEventControllerScroll *controller,
               gpointer                  user_data)
```

Signals that a new scrolling operation has finished. It will only be emitted on devices capable of it.

Parameters

controller	The object that received the signal
user_data	user data set when the signal handler was connected.

Flags: Run First

See Also

GtkEventController

GtkEventControllerMotion

GtkEventControllerMotion — Event controller for motion events

Functions

GtkEventController *	gtk_event_controller_motion_new ()

Signals

void	enter	Run First
void	leave	Run First
void	motion	Run First

Types and Values

GtkEventControllerMotion

Object Hierarchy

```
GObject
└── GtkEventController
    └── GtkEventControllerMotion
```

Includes

```
#include <gtk/gtk.h>
```

Description

GtkEventControllerMotion is an event controller meant for situations where you need to track the position of the pointer.

This object was added in 3.24.

Functions

gtk_event_controller_motion_new ()

```
GtkEventController *
gtk_event_controller_motion_new (GtkWidget *widget);
```

Creates a new event controller that will handle motion events for the given `widget` .

Parameters

widget	a GtkWidget

Returns

a new GtkEventControllerMotion

Since: 3.24

Types and Values

GtkEventControllerMotion

```
typedef struct _GtkEventControllerMotion GtkEventControllerMotion;
```

Signal Details

The `"enter"` signal

```
void
user_function (GtkEventControllerMotion *controller,
               gdouble                   x,
               gdouble                   y,
               gpointer                  user_data)
```

Signals that the pointer has entered the widget.

Parameters

controller	The object that received the signal
x	the x coordinate
y	the y coordinate
user_data	user data set when the signal handler was connected.

Flags: Run First

The "leave" signal

```
void
user_function (GtkEventControllerMotion *controller,
               gpointer                  user_data)
```

Signals that pointer has left the widget.

Parameters

controller	The object that received the signal
user_data	user data set when the signal handler was connected.

Flags: Run First

The "motion" signal

```
void
user_function (GtkEventControllerMotion *controller,
               gdouble                   x,
               gdouble                   y,
               gpointer                  user_data)
```

Emitted when the pointer moves inside the widget.

Parameters

controller	The object that received the signal
x	the x coordinate
y	the y coordinate
user_data	user data set when the signal handler was connected.

Flags: Run First

See Also

GtkEventController

GtkGesture

GtkGesture — Base class for gestures

Functions

GdkDevice *	gtk_gesture_get_device ()
GdkWindow *	gtk_gesture_get_window ()
void	gtk_gesture_set_window ()
gboolean	gtk_gesture_is_active ()

gboolean	gtk_gesture_is_recognized ()
GtkEventSequenceState	gtk_gesture_get_sequence_state ()
gboolean	gtk_gesture_set_sequence_state ()
gboolean	gtk_gesture_set_state ()
GList *	gtk_gesture_get_sequences ()
gboolean	gtk_gesture_handles_sequence ()
GdkEventSequence *	gtk_gesture_get_last_updated_sequence ()
const GdkEvent *	gtk_gesture_get_last_event ()
gboolean	gtk_gesture_get_point ()
gboolean	gtk_gesture_get_bounding_box ()
gboolean	gtk_gesture_get_bounding_box_center ()
void	gtk_gesture_group ()
void	gtk_gesture_ungroup ()
GList *	gtk_gesture_get_group ()
gboolean	gtk_gesture_is_grouped_with ()

Properties

guint	n-points	Read / Write / Construct Only
GdkWindow *	window	Read / Write

Signals

void	begin	Run Last
void	cancel	Run Last
void	end	Run Last
void	sequence-state-changed	Run Last
void	update	Run Last

Types and Values

	GtkGesture
enum	GtkEventSequenceState

Object Hierarchy

```
GObject
└── GtkEventController
    └── GtkGesture
        ├── GtkGestureSingle
        ├── GtkGestureRotate
        └── GtkGestureZoom
```

Includes

```
#include <gtk/gtk.h>
```

Description

GtkGesture is the base object for gesture recognition, although this object is quite generalized to serve as a base

for multi-touch gestures, it is suitable to implement single-touch and pointer-based gestures (using the special `NULL` GdkEventSequence value for these).

The number of touches that a GtkGesture need to be recognized is controlled by the “n-points” property, if a gesture is keeping track of less or more than that number of sequences, it won't check wether the gesture is recognized.

As soon as the gesture has the expected number of touches, the gesture will run the “check” signal regularly on input events until the gesture is recognized, the criteria to consider a gesture as "recognized" is left to GtkGesture subclasses.

A recognized gesture will then emit the following signals:

- “begin” when the gesture is recognized.
- A number of “update”, whenever an input event is processed.
- “end” when the gesture is no longer recognized.

Event propagation

In order to receive events, a gesture needs to either set a propagation phase through `gtk_event_controller_set_propagation_phase()`, or feed those manually through `gtk_event_controller_handle_event()`.

In the capture phase, events are propagated from the toplevel down to the target widget, and gestures that are attached to containers above the widget get a chance to interact with the event before it reaches the target.

After the capture phase, GTK+ emits the traditional “button-press-event”, “button-release-event”, “touch-event”, etc signals. Gestures with the `GTK_PHASE_TARGET` phase are fed events from the default “event” handlers.

In the bubble phase, events are propagated up from the target widget to the toplevel, and gestures that are attached to containers above the widget get a chance to interact with events that have not been handled yet.

States of a sequence

Whenever input interaction happens, a single event may trigger a cascade of GtkGestures, both across the parents of the widget receiving the event and in parallel within an individual widget. It is a responsibility of the widgets using those gestures to set the state of touch sequences accordingly in order to enable cooperation of gestures around the GdkEventSequences triggering those.

Within a widget, gestures can be grouped through `gtk_gesture_group()`, grouped gestures synchronize the state of sequences, so calling `gtk_gesture_set_sequence_state()` on one will effectively propagate the state throughout the group.

By default, all sequences start out in the GTK_EVENT_SEQUENCE_NONE state, sequences in this state trigger the gesture event handler, but event propagation will continue unstopped by gestures.

If a sequence enters into the GTK_EVENT_SEQUENCE_DENIED state, the gesture group will effectively ignore the sequence, letting events go unstopped through the gesture, but the "slot" will still remain occupied while the touch is active.

If a sequence enters in the GTK_EVENT_SEQUENCE_CLAIMED state, the gesture group will grab all interaction on the sequence, by:

- Setting the same sequence to GTK_EVENT_SEQUENCE_DENIED on every other gesture group within the widget, and every gesture on parent widgets in the propagation chain.
- calling “cancel” on every gesture in widgets underneath in the propagation chain.
- Stopping event propagation after the gesture group handles the event.

Note: if a sequence is set early to GTK_EVENT_SEQUENCE_CLAIMED on GDK_TOUCH_BEGIN/GDK_BUTTON_PRESS (so those events are captured before reaching the event widget, this implies GTK_PHASE_CAPTURE), one similar event will emulated if the sequence changes to GTK_EVENT_SEQUENCE_DENIED. This way event coherence is preserved before event propagation is unstopped again.

Sequence states can't be changed freely, see `gtk_gesture_set_sequence_state()` to know about the possible lifetimes of a GdkEventSequence.

Touchpad gestures

On the platforms that support it, GtkGesture will handle transparently touchpad gesture events. The only precautions users of GtkGesture should do to enable this support are:

- Enabling `GDK_TOUCHPAD_GESTURE_MASK` on their GdkWindows
- If the gesture has `GTK_PHASE_NONE`, ensuring events of type `GDK_TOUCHPAD_SWIPE` and `GDK_TOUCHPAD_PINCH` are handled by the GtkGesture

Functions

gtk_gesture_get_device ()

```
GdkDevice *
gtk_gesture_get_device (GtkGesture *gesture);
```

Returns the master GdkDevice that is currently operating on `gesture` , or `NULL` if the gesture is not being interacted.

Parameters

gesture	a GtkGesture

Returns

a GdkDevice, or `NULL`.

[nullable][transfer none]

Since: 3.14

---

gtk_gesture_get_window ()

```
GdkWindow *
gtk_gesture_get_window (GtkGesture *gesture);
```

Returns the user-defined window that receives the events handled by `gesture` . See `gtk_gesture_set_window()` for more information.

Parameters

gesture	a GtkGesture	

Returns

the user defined window, or `NULL` if none.

[nullable][transfer none]

Since: 3.14

---

gtk_gesture_set_window ()

```
void
gtk_gesture_set_window (GtkGesture *gesture,
                        GdkWindow *window);
```

Sets a specific window to receive events about, so `gesture` will effectively handle only events targeting `window` , or a child of it. `window` must pertain to `gtk_event_controller_get_widget()`.

Parameters

gesture	a GtkGesture	
window	a GdkWindow, or `NULL`.	[allow-none]

Since: 3.14

---

gtk_gesture_is_active ()

```
gboolean
gtk_gesture_is_active (GtkGesture *gesture);
```

Returns `TRUE` if the gesture is currently active. A gesture is active meanwhile there are touch sequences interacting with it.

Parameters

gesture	a GtkGesture	

Returns

TRUE if gesture is active

Since: 3.14

---

gtk_gesture_is_recognized ()

```
gboolean
gtk_gesture_is_recognized (GtkGesture *gesture);
```

Returns TRUE if the gesture is currently recognized. A gesture is recognized if there are as many interacting touch sequences as required by gesture , and “check” returned TRUE for the sequences being currently interpreted.

Parameters

gesture	a GtkGesture

Returns

TRUE if gesture is recognized

Since: 3.14

---

gtk_gesture_get_sequence_state ()

```
GtkEventSequenceState
gtk_gesture_get_sequence_state (GtkGesture *gesture,
                                GdkEventSequence *sequence);
```

Returns the sequence state, as seen by gesture .

Parameters

gesture	a GtkGesture
sequence	a GdkEventSequence

Returns

The sequence state in gesture

Since: 3.14

---

gtk_gesture_set_sequence_state ()

```
gboolean
gtk_gesture_set_sequence_state (GtkGesture *gesture,
```

```
                                GdkEventSequence *sequence,
                                GtkEventSequenceState state);
```

Sets the state of `sequence` in `gesture` . Sequences start in state GTK_EVENT_SEQUENCE_NONE, and whenever they change state, they can never go back to that state. Likewise, sequences in state GTK_EVENT_SEQUENCE_DENIED cannot turn back to a not denied state. With these rules, the lifetime of an event sequence is constrained to the next four:

- None
- None → Denied
- None → Claimed
- None → Claimed → Denied

Note: Due to event handling ordering, it may be unsafe to set the state on another gesture within a "begin" signal handler, as the callback might be executed before the other gesture knows about the sequence. A safe way to perform this could be:

```
static void
first_gesture_begin_cb (GtkGesture
*first_gesture,
                        GdkEventSequence
*sequence,
                        gpointer
user_data)
{
  gtk_gesture_set_sequence_state
(first_gesture, sequence,
GTK_EVENT_SEQUENCE_CLAIMED);
  gtk_gesture_set_sequence_state
(second_gesture, sequence,
GTK_EVENT_SEQUENCE_DENIED);
}

static void
second_gesture_begin_cb (GtkGesture
*second_gesture,
                         GdkEventSequence
*sequence,
                         gpointer
user_data)
{
  if (gtk_gesture_get_sequence_state
(first_gesture, sequence) ==
GTK_EVENT_SEQUENCE_CLAIMED)
    gtk_gesture_set_sequence_state
(second_gesture, sequence,
GTK_EVENT_SEQUENCE_DENIED);
}
```

If both gestures are in the same group, just set the state on the gesture emitting the event, the sequence will be already be initialized to the group's global state when the second gesture processes the event.

Parameters

gesture	a GtkGesture
sequence	a GdkEventSequence
state	the sequence state

Returns

TRUE if sequence is handled by gesture , and the state is changed successfully

Since: 3.14

---

gtk_gesture_set_state ()

```
gboolean
gtk_gesture_set_state (GtkGesture *gesture,
                       GtkEventSequenceState state);
```

Sets the state of all sequences that gesture is currently interacting with. See gtk_gesture_set_sequence_state() for more details on sequence states.

Parameters

gesture	a GtkGesture
state	the sequence state

Returns

TRUE if the state of at least one sequence was changed successfully

Since: 3.14

---

gtk_gesture_get_sequences ()

```
GList *
gtk_gesture_get_sequences (GtkGesture *gesture);
```

Returns the list of GdkEventSequences currently being interpreted by gesture .

Parameters

gesture	a GtkGesture

Returns

A list of GdkEventSequences, the list elements are owned by GTK+ and must not be freed or modified, the list itself must be deleted through g_list_free().

[transfer container][element-type GdkEventSequence]

Since: 3.14

---

gtk_gesture_handles_sequence ()

```
gboolean
gtk_gesture_handles_sequence (GtkGesture *gesture,
                              GdkEventSequence *sequence);
```

Returns TRUE if gesture is currently handling events corresponding to sequence .

Parameters

gesture	a GtkGesture	
sequence	a GdkEventSequence or NULL.	[nullable]

Returns

TRUE if gesture is handling sequence , FALSE otherwise

Since: 3.14

---

gtk_gesture_get_last_updated_sequence ()

```
GdkEventSequence *
gtk_gesture_get_last_updated_sequence (GtkGesture *gesture);
```

Returns the GdkEventSequence that was last updated on gesture .

Parameters

gesture	a GtkGesture

Returns

The last updated sequence.

[transfer none][nullable]

Since: 3.14

---

gtk_gesture_get_last_event ()

```
const GdkEvent *
gtk_gesture_get_last_event (GtkGesture *gesture,
                            GdkEventSequence *sequence);
```

Returns the last event that was processed for sequence .

Note that the returned pointer is only valid as long as the sequence is still interpreted by the gesture . If in doubt, you should make a copy of the event.

Parameters

gesture	a GtkGesture	
sequence	a GdkEventSequence.	[nullable]

Returns

The last event from `sequence` .

[transfer none][nullable]

---

gtk_gesture_get_point ()

```
gboolean
gtk_gesture_get_point (GtkGesture *gesture,
                       GdkEventSequence *sequence,
                       gdouble *x,
                       gdouble *y);
```

If `sequence` is currently being interpreted by `gesture` , this function returns `TRUE` and fills in `x` and `y` with the last coordinates stored for that event sequence. The coordinates are always relative to the widget allocation.

Parameters

gesture	a GtkGesture	
sequence	a GdkEventSequence, or `NULL` for pointer events.	[allow-none]
x	return location for X axis of the sequence coordinates.	[out][allow-none]
y	return location for Y axis of the sequence coordinates.	[out][allow-none]

Returns

`TRUE` if `sequence` is currently interpreted

Since: 3.14

---

gtk_gesture_get_bounding_box ()

```
gboolean
gtk_gesture_get_bounding_box (GtkGesture *gesture,
                              GdkRectangle *rect);
```

If there are touch sequences being currently handled by `gesture` , this function returns `TRUE` and fills in `rect` with the bounding box containing all active touches. Otherwise, `FALSE` will be returned.

Note: This function will yield unexpected results on touchpad gestures. Since there is no correlation between physical and pixel distances, these will look as if constrained in an infinitely small area, `rect` width and height will thus be 0 regardless of the number of touchpoints.

Parameters

gesture	a GtkGesture	
rect	bounding box containing all active touches.	[out]

Returns

`TRUE` if there are active touches, `FALSE` otherwise

Since: 3.14

---

gtk_gesture_get_bounding_box_center ()

```
gboolean
gtk_gesture_get_bounding_box_center (GtkGesture *gesture,
                                     gdouble *x,
                                     gdouble *y);
```

If there are touch sequences being currently handled by `gesture` , this function returns `TRUE` and fills in `x` and `y` with the center of the bounding box containing all active touches. Otherwise, `FALSE` will be returned.

Parameters

gesture	a GtkGesture	
x	X coordinate for the bounding box center.	[out]
y	Y coordinate for the bounding box center.	[out]

Returns

`FALSE` if no active touches are present, `TRUE` otherwise

Since: 3.14

---

gtk_gesture_group ()

```
void
gtk_gesture_group (GtkGesture *group_gesture,
                   GtkGesture *gesture);
```

Adds `gesture` to the same group than `group_gesture` . Gestures are by default isolated in their own groups.

When gestures are grouped, the state of GdkEventSequences is kept in sync for all of those, so calling `gtk_gesture_set_sequence_state()`, on one will transfer the same value to the others.

Groups also perform an "implicit grabbing" of sequences, if a GdkEventSequence state is set to GTK_EVENT_SEQUENCE_CLAIMED on one group, every other gesture group attached to the same GtkWidget will switch the state for that sequence to GTK_EVENT_SEQUENCE_DENIED.

Parameters

gesture	a GtkGesture
group_gesture	GtkGesture to group `gesture` with

Since: 3.14

---

gtk_gesture_ungroup ()

```
void
gtk_gesture_ungroup (GtkGesture *gesture);
```

Separates `gesture` into an isolated group.

Parameters

gesture	a GtkGesture

Since: 3.14

---

gtk_gesture_get_group ()

```
GList *
gtk_gesture_get_group (GtkGesture *gesture);
```

Returns all gestures in the group of `gesture`

Parameters

gesture	a GtkGesture

Returns

The list of GtkGestures, free with `g_list_free()`.

[element-type GtkGesture][transfer container]

Since: 3.14

---

gtk_gesture_is_grouped_with ()

```
gboolean
gtk_gesture_is_grouped_with (GtkGesture *gesture,
                             GtkGesture *other);
```

Returns `TRUE` if both gestures pertain to the same group.

Parameters

gesture	a GtkGesture

other	another GtkGesture

Returns

whether the gestures are grouped

Since: 3.14

Types and Values

GtkGesture

```
typedef struct _GtkGesture GtkGesture;
```

---

enum GtkEventSequenceState

Describes the state of a GdkEventSequence in a GtkGesture.

Members

GTK_EVENT_SEQUENCE_NONE	The sequence is handled, but not grabbed.
GTK_EVENT_SEQUENCE_CLAIMED	The sequence is handled and grabbed.
GTK_EVENT_SEQUENCE_DENIED	The sequence is denied.

Since: 3.14

Property Details

The `"n-points"` property

```
  "n-points"                 guint
```

The number of touch points that trigger recognition on this gesture,

Flags: Read / Write / Construct Only

Allowed values: >= 1

Default value: 1

Since: 3.14

The “window” property

```
  “window”                   GdkWindow *
```

If non-NULL, the gesture will only listen for events that happen on this GdkWindow, or a child of it.

Flags: Read / Write

Since: 3.14

Signal Details

The “begin” signal

```
void
user_function (GtkGesture       *gesture,
               GdkEventSequence *sequence,
               gpointer          user_data)
```

This signal is emitted when the gesture is recognized. This means the number of touch sequences matches “n-points”, and the “check” handler(s) returned TRUE.

Note: These conditions may also happen when an extra touch (eg. a third touch on a 2-touches gesture) is lifted, in that situation sequence won't pertain to the current set of active touches, so don't rely on this being true.

Parameters

gesture	the object which received the signal
sequence	the GdkEventSequence that made the gesture to be recognized
user_data	user data set when the signal handler was connected.

Flags: Run Last

Since: 3.14

The “cancel” signal

```
void
user_function (GtkGesture       *gesture,
               GdkEventSequence *sequence,
               gpointer          user_data)
```

This signal is emitted whenever a sequence is cancelled. This usually happens on active touches when gtk_event_controller_reset() is called on gesture (manually, due to grabs...), or the individual sequence was claimed by parent widgets' controllers (see gtk_gesture_set_sequence_state()).

gesture must forget everything about sequence as a reaction to this signal.

Parameters

gesture	the object which received the signal

sequence	the GdkEventSequence that was cancelled
user_data	user data set when the signal handler was connected.

Flags: Run Last

Since: 3.14

The “end” signal

```
void
user_function (GtkGesture       *gesture,
               GdkEventSequence *sequence,
               gpointer          user_data)
```

This signal is emitted when `gesture` either stopped recognizing the event sequences as something to be handled (the “check” handler returned `FALSE`), or the number of touch sequences became higher or lower than “n-points”.

Note: `sequence` might not pertain to the group of sequences that were previously triggering recognition on `gesture` (ie. a just pressed touch sequence that exceeds “n-points”). This situation may be detected by checking through `gtk_gesture_handles_sequence()`.

Parameters

gesture	the object which received the signal
sequence	the GdkEventSequence that made gesture recognition to finish
user_data	user data set when the signal handler was connected.

Flags: Run Last

Since: 3.14

The “sequence-state-changed” signal

```
void
user_function (GtkGesture           *gesture,
               GdkEventSequence     *sequence,
               GtkEventSequenceState state,
               gpointer              user_data)
```

This signal is emitted whenever a sequence state changes. See `gtk_gesture_set_sequence_state()` to know more about the expectable sequence lifetimes.

Parameters

gesture	the object which received the signal
sequence	the GdkEventSequence that was cancelled

state	the new sequence state
user_data	user data set when the signal handler was connected.

Flags: Run Last

Since: 3.14

The `"update"` signal

```
void
user_function (GtkGesture       *gesture,
               GdkEventSequence *sequence,
               gpointer          user_data)
```

This signal is emitted whenever an event is handled while the gesture is recognized. `sequence` is guaranteed to pertain to the set of active touches.

Parameters

gesture	the object which received the signal
sequence	the GdkEventSequence that was updated
user_data	user data set when the signal handler was connected.

Flags: Run Last

Since: 3.14

See Also

GtkEventController, GtkGestureSingle

GtkGestureSingle

GtkGestureSingle — Base class for mouse/single-touch gestures

Functions

gboolean	gtk_gesture_single_get_exclusive ()
void	gtk_gesture_single_set_exclusive ()
gboolean	gtk_gesture_single_get_touch_only ()
void	gtk_gesture_single_set_touch_only ()
guint	gtk_gesture_single_get_button ()
void	gtk_gesture_single_set_button ()
guint	gtk_gesture_single_get_current_button ()
GdkEventSequence *	gtk_gesture_single_get_current_sequence ()

Properties

guint	button	Read / Write
gboolean	exclusive	Read / Write
gboolean	touch-only	Read / Write

Types and Values

GtkGestureSingle

Object Hierarchy

```
GObject
└── GtkEventController
    └── GtkGesture
        └── GtkGestureSingle
            ├── GtkGestureDrag
            ├── GtkGestureLongPress
            ├── GtkGestureMultiPress
            ├── GtkGestureStylus
            └── GtkGestureSwipe
```

Includes

```
#include <gtk/gtk.h>
```

Description

GtkGestureSingle is a subclass of GtkGesture, optimized (although not restricted) for dealing with mouse and single-touch gestures. Under interaction, these gestures stick to the first interacting sequence, which is accessible through `gtk_gesture_single_get_current_sequence()` while the gesture is being interacted with.

By default gestures react to both `GDK_BUTTON_PRIMARY` and touch events, `gtk_gesture_single_set_touch_only()` can be used to change the touch behavior. Callers may also specify a different mouse button number to interact with through `gtk_gesture_single_set_button()`, or react to any mouse button by setting 0. While the gesture is active, the button being currently pressed can be known through `gtk_gesture_single_get_current_button()`.

Functions

gtk_gesture_single_get_exclusive ()

```
gboolean
gtk_gesture_single_get_exclusive (GtkGestureSingle *gesture);
```

Gets whether a gesture is exclusive. For more information, see `gtk_gesture_single_set_exclusive()`.

Parameters

gesture	a GtkGestureSingle

Returns

Whether the gesture is exclusive

Since: 3.14

---

gtk_gesture_single_set_exclusive ()

```
void
gtk_gesture_single_set_exclusive (GtkGestureSingle *gesture,
                                  gboolean exclusive);
```

Sets whether `gesture` is exclusive. An exclusive gesture will only handle pointer and "pointer emulated" touch events, so at any given time, there is only one sequence able to interact with those.

Parameters

gesture	a GtkGestureSingle
exclusive	`TRUE` to make `gesture` exclusive

Since: 3.14

---

gtk_gesture_single_get_touch_only ()

```
gboolean
gtk_gesture_single_get_touch_only (GtkGestureSingle *gesture);
```

Returns `TRUE` if the gesture is only triggered by touch events.

Parameters

gesture	a GtkGestureSingle

Returns

`TRUE` if the gesture only handles touch events

Since: 3.14

---

gtk_gesture_single_set_touch_only ()

```
void
gtk_gesture_single_set_touch_only (GtkGestureSingle *gesture,
                                   gboolean touch_only);
```

If `touch_only` is `TRUE`, `gesture` will only handle events of type GDK_TOUCH_BEGIN,

GDK_TOUCH_UPDATE or GDK_TOUCH_END. If FALSE, mouse events will be handled too.

Parameters

gesture	a GtkGestureSingle
touch_only	whether gesture handles only touch events

Since: 3.14

---

gtk_gesture_single_get_button ()

```
guint
gtk_gesture_single_get_button (GtkGestureSingle *gesture);
```

Returns the button number gesture listens for, or 0 if gesture reacts to any button press.

Parameters

gesture	a GtkGestureSingle

Returns

The button number, or 0 for any button

Since: 3.14

---

gtk_gesture_single_set_button ()

```
void
gtk_gesture_single_set_button (GtkGestureSingle *gesture,
                               guint button);
```

Sets the button number gesture listens to. If non-0, every button press from a different button number will be ignored. Touch events implicitly match with button 1.

Parameters

gesture	a GtkGestureSingle
button	button number to listen to, or 0 for any button

Since: 3.14

---

gtk_gesture_single_get_current_button ()

```
guint
gtk_gesture_single_get_current_button (GtkGestureSingle *gesture);
```

Returns the button number currently interacting with `gesture` , or 0 if there is none.

Parameters

gesture	a GtkGestureSingle

Returns

The current button number

Since: 3.14

---

gtk_gesture_single_get_current_sequence ()

```
GdkEventSequence *
gtk_gesture_single_get_current_sequence
                                (GtkGestureSingle *gesture);
```

Returns the event sequence currently interacting with `gesture` . This is only meaningful if `gtk_gesture_is_active()` returns `TRUE`.

Parameters

gesture	a GtkGestureSingle

Returns

the current sequence.

[nullable]

Since: 3.14

Types and Values

GtkGestureSingle

```
typedef struct _GtkGestureSingle GtkGestureSingle;
```

Property Details

The `“button”` property

```
  “button”                   guint
```

Mouse button number to listen to, or 0 to listen for any button.

Flags: Read / Write

Default value: 1

Since: 3.14

The “exclusive” property

“exclusive” gboolean

Whether the gesture is exclusive. Exclusive gestures only listen to pointer and pointer emulated events.

Flags: Read / Write

Default value: FALSE

Since: 3.14

The “touch-only” property

“touch-only” gboolean

Whether the gesture handles only touch events.

Flags: Read / Write

Default value: FALSE

Since: 3.14

GtkGestureDrag

GtkGestureDrag — Drag gesture

Functions

GtkGesture *	gtk_gesture_drag_new ()
gboolean	gtk_gesture_drag_get_start_point ()
gboolean	gtk_gesture_drag_get_offset ()

Signals

void	drag-begin	Run Last
void	drag-end	Run Last
void	drag-update	Run Last

Types and Values

GtkGestureDrag

Object Hierarchy

```
GObject
└── GtkEventController
    └── GtkGesture
        └── GtkGestureSingle
            └── GtkGestureDrag
                └── GtkGesturePan
```

Includes

```
#include <gtk/gtk.h>
```

Description

GtkGestureDrag is a GtkGesture implementation that recognizes drag operations. The drag operation itself can be tracked throught the “drag-begin”, “drag-update” and “drag-end” signals, or the relevant coordinates be extracted through `gtk_gesture_drag_get_offset()` and `gtk_gesture_drag_get_start_point()`.

Functions

gtk_gesture_drag_new ()

```
GtkGesture *
gtk_gesture_drag_new (GtkWidget *widget);
```

Returns a newly created GtkGesture that recognizes drags.

Parameters

widget	a GtkWidget

Returns

a newly created GtkGestureDrag

Since: 3.14

---

gtk_gesture_drag_get_start_point ()

```
gboolean
gtk_gesture_drag_get_start_point (GtkGestureDrag *gesture,
                                  gdouble *x,
                                  gdouble *y);
```

If the `gesture` is active, this function returns `TRUE` and fills in `x` and `y` with the drag start coordinates, in window-relative coordinates.

Parameters

gesture	a GtkGesture	
x	X coordinate for the drag start point.	[out][nullable]
y	Y coordinate for the drag start point.	[out][nullable]

Returns

`TRUE` if the gesture is active

Since: 3.14

---

gtk_gesture_drag_get_offset ()

```
gboolean
gtk_gesture_drag_get_offset (GtkGestureDrag *gesture,
                             gdouble *x,
                             gdouble *y);
```

If the `gesture` is active, this function returns `TRUE` and fills in `x` and `y` with the coordinates of the current point, as an offset to the starting drag point.

Parameters

gesture	a GtkGesture	
x	X offset for the current point.	[out][nullable]
y	Y offset for the current point.	[out][nullable]

Returns

`TRUE` if the gesture is active

Since: 3.14

Types and Values

GtkGestureDrag

```
typedef struct _GtkGestureDrag GtkGestureDrag;
```

Signal Details

The “drag-begin” signal

```
void
user_function (GtkGestureDrag *gesture,
               gdouble         start_x,
```

```
                gdouble         start_y,
                gpointer        user_data)
```

This signal is emitted whenever dragging starts.

Parameters

gesture	the object which received the signal
start_x	X coordinate, relative to the widget allocation
start_y	Y coordinate, relative to the widget allocation
user_data	user data set when the signal handler was connected.

Flags: Run Last

Since: 3.14

The `"drag-end"` signal

```
void
user_function (GtkGestureDrag *gesture,
               gdouble         offset_x,
               gdouble         offset_y,
               gpointer        user_data)
```

This signal is emitted whenever the dragging is finished.

Parameters

gesture	the object which received the signal
offset_x	X offset, relative to the start point
offset_y	Y offset, relative to the start point
user_data	user data set when the signal handler was connected.

Flags: Run Last

Since: 3.14

The `"drag-update"` signal

```
void
user_function (GtkGestureDrag *gesture,
               gdouble         offset_x,
               gdouble         offset_y,
               gpointer        user_data)
```

This signal is emitted whenever the dragging point moves.

Parameters

gesture	the object which received the signal
offset_x	X offset, relative to the start point
offset_y	Y offset, relative to the start point
user_data	user data set when the signal handler was connected.

Flags: Run Last

Since: 3.14

See Also

GtkGestureSwipe

GtkGestureLongPress

GtkGestureLongPress — "Press and Hold" gesture

Functions

GtkGesture *	gtk_gesture_long_press_new ()

Properties

gdouble	delay-factor	Read / Write

Signals

void	cancelled	Run Last
void	pressed	Run Last

Types and Values

GtkGestureLongPress

Object Hierarchy

```
GObject
└── GtkEventController
    └── GtkGesture
        └── GtkGestureSingle
            └── GtkGestureLongPress
```

Includes

```
#include <gtk/gtk.h>
```

Description

GtkGestureLongPress is a GtkGesture implementation able to recognize long presses, triggering the “pressed” after the timeout is exceeded.

If the touchpoint is lifted before the timeout passes, or if it drifts too far of the initial press point, the “cancelled” signal will be emitted.

Functions

gtk_gesture_long_press_new ()

```
GtkGesture *
gtk_gesture_long_press_new (GtkWidget *widget);
```

Returns a newly created GtkGesture that recognizes long presses.

Parameters

widget	a GtkWidget

Returns

a newly created GtkGestureLongPress

Since: 3.14

Types and Values

GtkGestureLongPress

```
typedef struct _GtkGestureLongPress GtkGestureLongPress;
```

Property Details

The “delay-factor” property

“delay-factor”	gdouble

Factor by which to modify the default timeout.

Flags: Read / Write

Allowed values: [0.5,2]

Default value: 1

Signal Details

The “cancelled” signal

```
void
user_function (GtkGestureLongPress *gesture,
               gpointer             user_data)
```

This signal is emitted whenever a press moved too far, or was released before “pressed” happened.

Parameters

gesture	the object which received the signal
user_data	user data set when the signal handler was connected.

Flags: Run Last

Since: 3.14

The “pressed” signal

```
void
user_function (GtkGestureLongPress *gesture,
               gdouble              x,
               gdouble              y,
               gpointer             user_data)
```

This signal is emitted whenever a press goes unmoved/unreleased longer than what the GTK+ defaults tell.

Parameters

gesture	the object which received the signal
x	the X coordinate where the press happened, relative to the widget allocation
y	the Y coordinate where the press happened, relative to the widget allocation
user_data	user data set when the signal handler was connected.

Flags: Run Last

Since: 3.14

GtkGestureMultiPress

GtkGestureMultiPress — Multipress gesture

Functions

GtkGesture *	gtk_gesture_multi_press_new ()
void	gtk_gesture_multi_press_set_area ()
gboolean	gtk_gesture_multi_press_get_area ()

Signals

void	pressed	Run Last
void	released	Run Last
void	stopped	Run Last

Types and Values

GtkGestureMultiPress

Object Hierarchy

```
GObject
└── GtkEventController
    └── GtkGesture
        └── GtkGestureSingle
            └── GtkGestureMultiPress
```

Includes

```
#include <gtk/gtk.h>
```

Description

GtkGestureMultiPress is a GtkGesture implementation able to recognize multiple clicks on a nearby zone, which can be listened for through the “pressed” signal. Whenever time or distance between clicks exceed the GTK+ defaults, “stopped” is emitted, and the click counter is reset.

Callers may also restrict the area that is considered valid for a >1 touch/button press through `gtk_gesture_multi_press_set_area()`, so any click happening outside that area is considered to be a first click of its own.

Functions

gtk_gesture_multi_press_new ()

```
GtkGesture *
gtk_gesture_multi_press_new (GtkWidget *widget);
```

Returns a newly created GtkGesture that recognizes single and multiple presses.

Parameters

widget	a GtkWidget

Returns

a newly created GtkGestureMultiPress

Since: 3.14

---

gtk_gesture_multi_press_set_area ()

```
void
gtk_gesture_multi_press_set_area (GtkGestureMultiPress *gesture,
                                  const GdkRectangle *rect);
```

If `rect` is non-`NULL`, the press area will be checked to be confined within the rectangle, otherwise the button count will be reset so the press is seen as being the first one. If `rect` is `NULL`, the area will be reset to an unrestricted state.

Note: The rectangle is only used to determine whether any non-first click falls within the expected area. This is not akin to an input shape.

Parameters

gesture	a GtkGestureMultiPress	
rect	rectangle to receive coordinates on.	[allow-none]

Since: 3.14

---

gtk_gesture_multi_press_get_area ()

```
gboolean
gtk_gesture_multi_press_get_area (GtkGestureMultiPress *gesture,
                                  GdkRectangle *rect);
```

If an area was set through `gtk_gesture_multi_press_set_area()`, this function will return `TRUE` and fill in `rect` with the press area. See `gtk_gesture_multi_press_set_area()` for more details on what the press area represents.

Parameters

gesture	a GtkGestureMultiPress	
rect	return location for the press area.	[out]

Returns

`TRUE` if `rect` was filled with the press area

Since: 3.14

Types and Values

GtkGestureMultiPress

```
typedef struct _GtkGestureMultiPress GtkGestureMultiPress;
```

Signal Details

The “pressed” signal

```
void
user_function (GtkGestureMultiPress *gesture,
               gint                  n_press,
               gdouble               x,
               gdouble               y,
               gpointer              user_data)
```

This signal is emitted whenever a button or touch press happens.

Parameters

gesture	the object which received the signal
n_press	how many touch/button presses happened with this one
x	The X coordinate, in widget allocation coordinates
y	The Y coordinate, in widget allocation coordinates
user_data	user data set when the signal handler was connected.

Flags: Run Last

Since: 3.14

The “released” signal

```
void
user_function (GtkGestureMultiPress *gesture,
               gint                  n_press,
               gdouble               x,
               gdouble               y,
               gpointer              user_data)
```

This signal is emitted when a button or touch is released. `n_press` will report the number of press that is paired to this event, note that “stopped” may have been emitted between the press and its release, `n_press` will only start over at the next press.

Parameters

gesture	the object which received the signal
n_press	number of press that is paired with this release
x	The X coordinate, in widget allocation coordinates
y	The Y coordinate, in widget allocation coordinates
user_data	user data set when the signal handler was connected.

Flags: Run Last

Since: 3.14

The `"stopped"` signal

```
void
user_function (GtkGestureMultiPress *gesture,
               gpointer              user_data)
```

This signal is emitted whenever any time/distance threshold has been exceeded.

Parameters

gesture	the object which received the signal
user_data	user data set when the signal handler was connected.

Flags: Run Last

Since: 3.14

GtkGesturePan

GtkGesturePan — Pan gesture

Functions

GtkGesture *	gtk_gesture_pan_new ()
GtkOrientation	gtk_gesture_pan_get_orientation ()
void	gtk_gesture_pan_set_orientation ()

Properties

GtkOrientation	orientation	Read / Write

Signals

void	pan	Run Last

Types and Values

	GtkGesturePan
enum	GtkPanDirection

Object Hierarchy

```
GObject
└── GtkEventController
    └── GtkGesture
        └── GtkGestureSingle
            └── GtkGestureDrag
                └── GtkGesturePan
```

Includes

```
#include <gtk/gtk.h>
```

Description

GtkGesturePan is a GtkGesture implementation able to recognize pan gestures, those are drags that are locked to happen along one axis. The axis that a GtkGesturePan handles is defined at construct time, and can be changed through `gtk_gesture_pan_set_orientation()`.

When the gesture starts to be recognized, GtkGesturePan will attempt to determine as early as possible whether the sequence is moving in the expected direction, and denying the sequence if this does not happen.

Once a panning gesture along the expected axis is recognized, the "pan" signal will be emitted as input events are received, containing the offset in the given axis.

Functions

gtk_gesture_pan_new ()

```
GtkGesture *
gtk_gesture_pan_new (GtkWidget *widget,
                     GtkOrientation orientation);
```

Returns a newly created GtkGesture that recognizes pan gestures.

Parameters

widget	a GtkWidget
orientation	expected orientation

Returns

a newly created GtkGesturePan

Since: 3.14

---

gtk_gesture_pan_get_orientation ()

```
GtkOrientation
gtk_gesture_pan_get_orientation (GtkGesturePan *gesture);
```

Returns the orientation of the pan gestures that this `gesture` expects.

Parameters

gesture	A GtkGesturePan

Returns

the expected orientation for pan gestures

Since: 3.14

---

gtk_gesture_pan_set_orientation ()

```
void
gtk_gesture_pan_set_orientation (GtkGesturePan *gesture,
                                 GtkOrientation orientation);
```

Sets the orientation to be expected on pan gestures.

Parameters

gesture	A GtkGesturePan
orientation	expected orientation

Since: 3.14

Types and Values

GtkGesturePan

```
typedef struct _GtkGesturePan GtkGesturePan;
```

---

enum GtkPanDirection

Describes the panning direction of a GtkGesturePan

Members

GTK_PAN_DIRECTION_LEFT	panned towards the left
GTK_PAN_DIRECTION_RIGHT	panned towards the right
GTK_PAN_DIRECTION_UP	panned upwards
GTK_PAN_DIRECTION_DOWN	panned downwards

Since: 3.14

Property Details

The `"orientation"` property

```
  "orientation"              GtkOrientation
```

The expected orientation of pan gestures.

Flags: Read / Write

Default value: GTK_ORIENTATION_HORIZONTAL

Since: 3.14

Signal Details

The `"pan"` signal

```
void
user_function (GtkGesturePan  *gesture,
               GtkPanDirection direction,
               gdouble         offset,
               gpointer        user_data)
```

This signal is emitted once a panning gesture along the expected axis is detected.

Parameters

gesture	The object which received the signal
direction	current direction of the pan gesture
offset	Offset along the gesture orientation
user_data	user data set when the signal handler was connected.

Flags: Run Last

Since: 3.14

GtkGestureSwipe

GtkGestureSwipe — Swipe gesture

Functions

GtkGesture *	gtk_gesture_swipe_new ()
gboolean	gtk_gesture_swipe_get_velocity ()

Signals

void	swipe	Run Last

Types and Values

GtkGestureSwipe

Object Hierarchy

```
GObject
└── GtkEventController
    └── GtkGesture
        └── GtkGestureSingle
            └── GtkGestureSwipe
```

Includes

```
#include <gtk/gtk.h>
```

Description

GtkGestureSwipe is a GtkGesture implementation able to recognize swipes, after a press/move/.../move/release sequence happens, the "swipe" signal will be emitted, providing the velocity and directionality of the sequence at the time it was lifted.

If the velocity is desired in intermediate points, `gtk_gesture_swipe_get_velocity()` can be called on eg. a "update" handler.

All velocities are reported in pixels/sec units.

Functions

gtk_gesture_swipe_new ()

```
GtkGesture *
gtk_gesture_swipe_new (GtkWidget *widget);
```

Returns a newly created GtkGesture that recognizes swipes.

Parameters

widget	a GtkWidget

Returns

a newly created GtkGestureSwipe

Since: 3.14

---

gtk_gesture_swipe_get_velocity ()

```
gboolean
gtk_gesture_swipe_get_velocity (GtkGestureSwipe *gesture,
                                gdouble *velocity_x,
                                gdouble *velocity_y);
```

If the gesture is recognized, this function returns `TRUE` and fill in `velocity_x` and `velocity_y` with the recorded velocity, as per the last event(s) processed.

Parameters

gesture	a GtkGestureSwipe	
velocity_x	return value for the velocity in the X axis, in pixels/sec.	[out]
velocity_y	return value for the velocity in the Y axis, in pixels/sec.	[out]

Returns

whether velocity could be calculated

Since: 3.14

Types and Values

GtkGestureSwipe

```
typedef struct _GtkGestureSwipe GtkGestureSwipe;
```

Signal Details

The “swipe” signal

```
void
```

```
user_function (GtkGestureSwipe *gesture,
               gdouble          velocity_x,
               gdouble          velocity_y,
               gpointer         user_data)
```

This signal is emitted when the recognized gesture is finished, velocity and direction are a product of previously recorded events.

Parameters

gesture	object which received the signal
velocity_x	velocity in the X axis, in pixels/sec
velocity_y	velocity in the Y axis, in pixels/sec
user_data	user data set when the signal handler was connected.

Flags: Run Last

Since: 3.14

GtkGestureRotate

GtkGestureRotate — Rotate gesture

Functions

GtkGesture *	gtk_gesture_rotate_new ()
gdouble	gtk_gesture_rotate_get_angle_delta ()

Signals

void	angle-changed	Run First

Types and Values

GtkGestureRotate

Object Hierarchy

```
GObject
└── GtkEventController
    └── GtkGesture
        └── GtkGestureRotate
```

Includes

```
#include <gtk/gtk.h>
```

Description

GtkGestureRotate is a GtkGesture implementation able to recognize 2-finger rotations, whenever the angle between both handled sequences changes, the “angle-changed” signal is emitted.

Functions

gtk_gesture_rotate_new ()

```
GtkGesture *
gtk_gesture_rotate_new (GtkWidget *widget);
```

Returns a newly created GtkGesture that recognizes 2-touch rotation gestures.

Parameters

widget	a GtkWidget

Returns

a newly created GtkGestureRotate

Since: 3.14

---

gtk_gesture_rotate_get_angle_delta ()

```
gdouble
gtk_gesture_rotate_get_angle_delta (GtkGestureRotate *gesture);
```

If `gesture` is active, this function returns the angle difference in radians since the gesture was first recognized. If `gesture` is not active, 0 is returned.

Parameters

gesture	a GtkGestureRotate

Returns

the angle delta in radians

Since: 3.14

Types and Values

GtkGestureRotate

```
typedef struct _GtkGestureRotate GtkGestureRotate;
```

Signal Details

The "angle-changed" signal

```
void
user_function (GtkGestureRotate *gesture,
               gdouble           angle,
               gdouble           angle_delta,
               gpointer          user_data)
```

This signal is emitted when the angle between both tracked points changes.

Parameters

gesture	the object on which the signal is emitted
angle	Current angle in radians
angle_delta	Difference with the starting angle, in radians
user_data	user data set when the signal handler was connected.

Flags: Run First

Since: 3.14

See Also

GtkGestureZoom

GtkGestureZoom

GtkGestureZoom — Zoom gesture

Functions

GtkGesture *	gtk_gesture_zoom_new ()
gdouble	gtk_gesture_zoom_get_scale_delta ()

Signals

void	scale-changed	Run First

Types and Values

GtkGestureZoom

Object Hierarchy

```
GObject
└── GtkEventController
    └── GtkGesture
        └── GtkGestureZoom
```

Includes

```
#include <gtk/gtk.h>
```

Description

GtkGestureZoom is a GtkGesture implementation able to recognize pinch/zoom gestures, whenever the distance between both tracked sequences changes, the "scale-changed" signal is emitted to report the scale factor.

Functions

gtk_gesture_zoom_new ()

```
GtkGesture *
gtk_gesture_zoom_new (GtkWidget *widget);
```

Returns a newly created GtkGesture that recognizes zoom in/out gestures (usually known as pinch/zoom).

Parameters

widget	a GtkWidget

Returns

a newly created GtkGestureZoom

Since: 3.14

---

gtk_gesture_zoom_get_scale_delta ()

```
gdouble
gtk_gesture_zoom_get_scale_delta (GtkGestureZoom *gesture);
```

If `gesture` is active, this function returns the zooming difference since the gesture was recognized (hence the starting point is considered 1:1). If `gesture` is not active, 1 is returned.

Parameters

gesture	a GtkGestureZoom

Returns

the scale delta

Since: 3.14

Types and Values

GtkGestureZoom

```
typedef struct _GtkGestureZoom GtkGestureZoom;
```

Signal Details

The "scale-changed" signal

```
void
user_function (GtkGestureZoom *controller,
               gdouble         scale,
               gpointer        user_data)
```

This signal is emitted whenever the distance between both tracked sequences changes.

Parameters

controller	the object on which the signal is emitted
scale	Scale delta, taking the initial state as 1:1
user_data	user data set when the signal handler was connected.

Flags: Run First

Since: 3.14

See Also

GtkGestureRotate

GtkGestureStylus

GtkGestureStylus — Gesture for stylus input

Functions

GtkGesture *	gtk_gesture_stylus_new ()
gboolean	gtk_gesture_stylus_get_axis ()
gboolean	gtk_gesture_stylus_get_axes ()
GdkDeviceTool *	gtk_gesture_stylus_get_device_tool ()

Signals

void	down	Run Last
void	motion	Run Last
void	proximity	Run Last
void	up	Run Last

Types and Values

GtkGestureStylus

Object Hierarchy

```
GObject
└── GtkEventController
    └── GtkGesture
        └── GtkGestureSingle
            └── GtkGestureStylus
```

Includes

```
#include <gtk/gtk.h>
```

Description

GtkGestureStylus is a GtkGesture implementation specific to stylus input. The provided signals just provide the basic information

Functions

gtk_gesture_stylus_new ()

```
GtkGesture *
gtk_gesture_stylus_new (GtkWidget *widget);
```

Creates a new GtkGestureStylus.

Parameters

widget	a GtkWidget

Returns

a newly created stylus gesture

Since: 3.24

---

gtk_gesture_stylus_get_axis ()

```
gboolean
gtk_gesture_stylus_get_axis (GtkGestureStylus *gesture,
                             GdkAxisUse axis,
                             gdouble *value);
```

Returns the current value for the requested `axis` . This function must be called from either the “down”, “motion”, “up” or “proximity” signals.

Parameters

gesture	a GtkGestureStylus	
axis	requested device axis	
value	return location for the axis value.	[out]

Returns

TRUE if there is a current value for the axis

Since: 3.24

---

gtk_gesture_stylus_get_axes ()

```
gboolean
gtk_gesture_stylus_get_axes (GtkGestureStylus *gesture,
                             GdkAxisUse axes[],
                             gdouble **values);
```

Returns the current values for the requested `axes` . This function must be called from either the “down”, “motion”, “up” or “proximity” signals.

Parameters

gesture	a GtkGestureStylus	
axes	array of requested axes, terminated with GDK_AXIS_IGNORE.	[array]
values	return location for the axis values.	[out][array]

Returns

TRUE if there is a current value for the axes

Since: 3.24

---

gtk_gesture_stylus_get_device_tool ()

```
GdkDeviceTool *
gtk_gesture_stylus_get_device_tool (GtkGestureStylus *gesture);
```

Returns the GdkDeviceTool currently driving input through this gesture. This function must be called from either the “down”, “motion”, “up” or “proximity” signal handlers.

Parameters

gesture	a GtkGestureStylus

Returns

The current stylus tool.

[nullable][transfer none]

Since: 3.24

Types and Values

GtkGestureStylus

```
typedef struct _GtkGestureStylus GtkGestureStylus;
```

Signal Details

The “down” signal

```
void
user_function (GtkGestureStylus *gesturestylus,
               gdouble           arg1,
               gdouble           arg2,
               gpointer          user_data)
```

Flags: Run Last

The “motion” signal

```
void
user_function (GtkGestureStylus *gesturestylus,
               gdouble           arg1,
               gdouble           arg2,
               gpointer          user_data)
```

Flags: Run Last

The “proximity” signal

```
void
user_function (GtkGestureStylus *gesturestylus,
               gdouble           arg1,
               gdouble           arg2,
               gpointer          user_data)
```

Flags: Run Last

The “up” signal

```
void
user_function (GtkGestureStylus *gesturestylus,
               gdouble           arg1,
               gdouble           arg2,
               gpointer          user_data)
```

Flags: Run Last

See Also

GtkGesture, GtkGestureSingle

GtkPadController

GtkPadController — Controller for drawing tablet pads

Functions

GtkPadController *	gtk_pad_controller_new ()
void	gtk_pad_controller_set_action_entries ()
void	gtk_pad_controller_set_action ()

Properties

GActionGroup *	action-group	Read / Write / Construct Only
GdkDevice *	pad	Read / Write / Construct Only

Types and Values

	GtkPadController
enum	GtkPadActionType
struct	GtkPadActionEntry

Object Hierarchy

```
GObject
└── GtkEventController
    └── GtkPadController
```

Includes

```
#include <gtk/gtk.h>
```

Description

GtkPadController is an event controller for the pads found in drawing tablets (The collection of buttons and tactile sensors often found around the stylus-sensitive area).

These buttons and sensors have no implicit meaning, and by default they perform no action, this event controller is provided to map those to GAction objects, thus letting the application give those a more semantic meaning.

Buttons and sensors are not constrained to triggering a single action, some `GDK_SOURCE_TABLET_PAD` devices feature multiple "modes", all these input elements have one current mode, which may determine the final action being triggered. Pad devices often divide buttons and sensors into groups, all elements in a group share the same current mode, but different groups may have different modes. See `gdk_device_pad_get_n_groups()` and `gdk_device_pad_get_group_n_modes()`.

Each of the actions that a given button/strip/ring performs for a given mode is defined by GtkPadActionEntry, it contains an action name that will be looked up in the given GActionGroup and activated whenever the specified input element and mode are triggered.

A simple example of GtkPadController usage, assigning button 1 in all modes and pad devices to an "invert-selection" action:

```
GtkPadActionEntry *pad_actions[] = {
  { GTK_PAD_ACTION_BUTTON, 1, -1, "Invert selection", "pad-actions.invert-selection" },
  …
};

…
action_group = g_simple_action_group_new ();
action = g_simple_action_new ("pad-actions.invert-selection", NULL);
g_signal_connect (action, "activate", on_invert_selection_activated, NULL);
g_action_map_add_action (G_ACTION_MAP (action_group), action);
…
pad_controller = gtk_pad_controller_new (window, action_group, NULL);
```

The actions belonging to rings/strips will be activated with a parameter of type `G_VARIANT_TYPE_DOUBLE` bearing the value of the given axis, it is required that those are made stateful and accepting this GVariantType.

Functions

gtk_pad_controller_new ()

```
GtkPadController *
gtk_pad_controller_new (GtkWindow *window,
                        GActionGroup *group,
                        GdkDevice *pad);
```

Creates a new GtkPadController that will associate events from `pad` to actions. A `NULL` pad may be provided so the controller manages all pad devices generically, it is discouraged to mix GtkPadController objects with `NULL` and non-`NULL` `pad` argument on the same `window` , as execution order is not guaranteed.

The GtkPadController is created with no mapped actions. In order to map pad events to actions, use `gtk_pad_controller_set_action_entries()` or `gtk_pad_controller_set_action()`.

Parameters

window	a GtkWindow	
group	GActionGroup to trigger actions from	
pad	A `GDK_SOURCE_TABLET_PAD` device, or `NULL` to handle all pads.	[nullable]

Returns

A newly created GtkPadController

Since: 3.22

---

gtk_pad_controller_set_action_entries ()

```
void
gtk_pad_controller_set_action_entries (GtkPadController *controller,
                                       const GtkPadActionEntry *entries,
                                       gint n_entries);
```

This is a convenience function to add a group of action entries on `controller` . See GtkPadActionEntry and `gtk_pad_controller_set_action()`.

Parameters

controller	a GtkPadController	
entries	the action entries to set on `controller` .	[array length=n_entries]
n_entries	the number of elements in `entries`	

Since: 3.22

---

gtk_pad_controller_set_action ()

```
void
gtk_pad_controller_set_action (GtkPadController *controller,
                               GtkPadActionType type,
                               gint index,
                               gint mode,
                               const gchar *label,
                               const gchar *action_name);
```

Adds an individual action to `controller` . This action will only be activated if the given button/ring/strip number in `index` is interacted while the current mode is `mode` . -1 may be used for simple cases, so the action is triggered on all modes.

The given `label` should be considered user-visible, so internationalization rules apply. Some windowing systems may be able to use those for user feedback.

Parameters

controller	a GtkPadController
type	the type of pad feature that will trigger this action
index	the 0-indexed button/ring/strip number that will trigger this action
mode	the mode that will trigger this action, or -1 for all modes.
label	Human readable description of this action, this string should be deemed user-visible.
action_name	action name that will be activated in the GActionGroup

Since: 3.22

Types and Values

GtkPadController

```
typedef struct _GtkPadController GtkPadController;
```

---

enum GtkPadActionType

The type of a pad action.

Members

GTK_PAD_ACTION_BUTTON	Action is triggered by a pad button
GTK_PAD_ACTION_RING	Action is triggered by a pad ring
GTK_PAD_ACTION_STRIP	Action is triggered by a pad strip

---

struct GtkPadActionEntry

```
struct GtkPadActionEntry {
  GtkPadActionType type;
  gint index;
  gint mode;
  gchar *label;
  gchar *action_name;
};
```

Struct defining a pad action entry.

Members

GtkPadActionType `type`;	the type of pad feature that will trigger this action entry.
gint `index`;	the 0-indexed button/ring/strip number that will trigger this action entry.
gint `mode`;	the mode that will trigger this action entry, or -1 for all modes.
gchar *`label`;	Human readable description of this action entry, this string should be deemed user-visible.
gchar *`action_name`;	action name that will be activated in the GActionGroup.

Property Details

The `"action-group"` property

```
  "action-group"             GActionGroup *
```

Action group to launch actions from.

Flags: Read / Write / Construct Only

The `"pad"` property

```
  "pad"                      GdkDevice *
```

Pad device to control.

Flags: Read / Write / Construct Only

See Also

GtkEventController, GdkDevicePad

Deprecated

GtkSymbolicColor — Symbolic colors

GtkGradient — Gradients

Resource Files — Deprecated routines for handling resource files

GtkStyle — Deprecated object that holds style information for widgets

GtkHScale — A horizontal slider widget for selecting a value from a range

GtkVScale — A vertical slider widget for selecting a value from a range

GtkTearoffMenuItem — A menu item used to tear off and reattach its menu

GtkColorSelection — Deprecated widget used to select a color

GtkColorSelectionDialog — Deprecated dialog box for selecting a color

GtkHSV — A “color wheel” widget

GtkFontSelection — Deprecated widget for selecting fonts

GtkFontSelectionDialog — Deprecated dialog box for selecting fonts

GtkHBox — A horizontal container box

GtkVBox — A vertical container box

GtkHButtonBox — A container for arranging buttons horizontally

GtkVButtonBox — A container for arranging buttons vertically

GtkHPaned — A container with two panes arranged horizontally

GtkVPaned — A container with two panes arranged vertically

GtkTable — Pack widgets in regular patterns

GtkHSeparator — A horizontal separator

GtkVSeparator — A vertical separator

GtkHScrollbar — A horizontal scrollbar

GtkVScrollbar — A vertical scrollbar

GtkUIManager — Constructing menus and toolbars from an XML description

GtkActionGroup — A group of actions

GtkAction — A deprecated action which can be triggered by a menu or toolbar item

GtkToggleAction — An action which can be toggled between two states

GtkRadioAction — An action of which only one in a group can be active

GtkRecentAction — An action of which represents a list of recently used files

GtkActivatable — An interface for activatable widgets

GtkImageMenuItem — A deprecated widget for a menu item with an icon

GtkMisc — Base class for widgets with alignments and padding

Stock Items — Prebuilt common menu/toolbar items and corresponding icons

Themeable Stock Images — Manipulating stock icons

GtkNumerableIcon — A GIcon that allows numbered emblems

GtkArrow — Displays an arrow

GtkStatusIcon — Display an icon in the system tray

GtkThemingEngine — Theming renderers

GtkAlignment — A widget which controls the alignment and size of its child

---

GtkSymbolicColor

GtkSymbolicColor — Symbolic colors

Functions

GtkSymbolicColor *	gtk_symbolic_color_new_literal ()
GtkSymbolicColor *	gtk_symbolic_color_new_name ()
GtkSymbolicColor *	gtk_symbolic_color_new_shade ()
GtkSymbolicColor *	gtk_symbolic_color_new_alpha ()
GtkSymbolicColor *	gtk_symbolic_color_new_mix ()
GtkSymbolicColor *	gtk_symbolic_color_new_win32 ()
GtkSymbolicColor *	gtk_symbolic_color_ref ()
void	gtk_symbolic_color_unref ()
gboolean	gtk_symbolic_color_resolve ()
char *	gtk_symbolic_color_to_string ()

Types and Values

GtkSymbolicColor

Includes

```
#include <gtk/gtk.h>
```

Description

GtkSymbolicColor is a boxed type that represents a symbolic color. It is the result of parsing a color expression. To obtain the color represented by a GtkSymbolicColor, it has to be resolved with `gtk_symbolic_color_resolve()`, which replaces all symbolic color references by the colors they refer to (in a given context) and evaluates mix, shade and other expressions, resulting in a GdkRGBA value.

It is not normally necessary to deal directly with GtkSymbolicColors, since they are mostly used behind the scenes by GtkStyleContext and GtkCssProvider.

GtkSymbolicColor is deprecated. Symbolic colors are considered an implementation detail of GTK+.

Functions

gtk_symbolic_color_new_literal ()

```
GtkSymbolicColor *
gtk_symbolic_color_new_literal (const GdkRGBA *color);
```

`gtk_symbolic_color_new_literal` has been deprecated since version 3.8 and should not be used in newly-written code.

GtkSymbolicColor is deprecated.

Creates a symbolic color pointing to a literal color.

Parameters

color	a GdkRGBA

Returns

a newly created GtkSymbolicColor

Since: 3.0

---

gtk_symbolic_color_new_name ()

```
GtkSymbolicColor *
gtk_symbolic_color_new_name (const gchar *name);
```

`gtk_symbolic_color_new_name` has been deprecated since version 3.8 and should not be used in newly-written code.

GtkSymbolicColor is deprecated.

Creates a symbolic color pointing to an unresolved named color. See `gtk_style_context_lookup_color()` and `gtk_style_properties_lookup_color()`.

Parameters

name	color name

Returns

a newly created GtkSymbolicColor

Since: 3.0

---

gtk_symbolic_color_new_shade ()

```
GtkSymbolicColor *
gtk_symbolic_color_new_shade (GtkSymbolicColor *color,
                              gdouble factor);
```

`gtk_symbolic_color_new_shade` has been deprecated since version 3.8 and should not be used in newly-written code.

GtkSymbolicColor is deprecated.

Creates a symbolic color defined as a shade of another color. A factor > 1.0 would resolve to a brighter color, while < 1.0 would resolve to a darker color.

[constructor]

Parameters

color	another GtkSymbolicColor
factor	shading factor to apply to `color`

Returns

A newly created GtkSymbolicColor

Since: 3.0

---

gtk_symbolic_color_new_alpha ()

```
GtkSymbolicColor *
gtk_symbolic_color_new_alpha (GtkSymbolicColor *color,
                              gdouble factor);
```

`gtk_symbolic_color_new_alpha` has been deprecated since version 3.8 and should not be used in newly-written code.

GtkSymbolicColor is deprecated.

Creates a symbolic color by modifying the relative alpha value of `color` . A factor < 1.0 would resolve to a more transparent color, while > 1.0 would resolve to a more opaque color.

[constructor]

Parameters

color	another GtkSymbolicColor
factor	factor to apply to `color` alpha

Returns

A newly created GtkSymbolicColor

Since: 3.0

---

gtk_symbolic_color_new_mix ()

```
GtkSymbolicColor *
gtk_symbolic_color_new_mix (GtkSymbolicColor *color1,
                            GtkSymbolicColor *color2,
                            gdouble factor);
```

`gtk_symbolic_color_new_mix` has been deprecated since version 3.8 and should not be used in newly-written code.

GtkSymbolicColor is deprecated.

Creates a symbolic color defined as a mix of another two colors. a mix factor of 0 would resolve to `color1` , while a factor of 1 would resolve to `color2` .

[constructor]

Parameters

color1	color to mix
color2	another color to mix
factor	mix factor

Returns

A newly created GtkSymbolicColor

Since: 3.0

---

gtk_symbolic_color_new_win32 ()

```
GtkSymbolicColor *
gtk_symbolic_color_new_win32 (const gchar *theme_class,
                              gint id);
```

`gtk_symbolic_color_new_win32` has been deprecated since version 3.8 and should not be used in newly-written code.

GtkSymbolicColor is deprecated.

Creates a symbolic color based on the current win32 theme.

Note that while this call is available on all platforms the actual value returned is not reliable on non-win32 platforms.

[constructor]

Parameters

theme_class	The theme class to pull color from
id	The color id

Returns

A newly created GtkSymbolicColor

Since: 3.4

---

gtk_symbolic_color_ref ()

```
GtkSymbolicColor *
gtk_symbolic_color_ref (GtkSymbolicColor *color);
```

`gtk_symbolic_color_ref` has been deprecated since version 3.8 and should not be used in newly-written code.

GtkSymbolicColor is deprecated.

Increases the reference count of `color`

Parameters

color	a GtkSymbolicColor

Returns

the same `color`

Since: 3.0

---

gtk_symbolic_color_unref ()

```
void
gtk_symbolic_color_unref (GtkSymbolicColor *color);
```

`gtk_symbolic_color_unref` has been deprecated since version 3.8 and should not be used in newly-written code.

GtkSymbolicColor is deprecated.

Decreases the reference count of `color` , freeing its memory if the reference count reaches 0.

Parameters

color	a GtkSymbolicColor

Since: 3.0

---

gtk_symbolic_color_resolve ()

```
gboolean
gtk_symbolic_color_resolve (GtkSymbolicColor *color,
                            GtkStyleProperties *props,
                            GdkRGBA *resolved_color);
```

`gtk_symbolic_color_resolve` has been deprecated since version 3.8 and should not be used in newly-written code.

GtkSymbolicColor is deprecated.

If `color` is resolvable, `resolved_color` will be filled in with the resolved color, and `TRUE` will be returned. Generally, if `color` can't be resolved, it is due to it being defined on top of a named color that doesn't exist in `props` .

When `props` is `NULL`, resolving of named colors will fail, so if your `color` is or references such a color, this function will return `FALSE`.

Parameters

color	a GtkSymbolicColor	
props	GtkStyleProperties to use when resolving named colors, or `NULL`.	[allow-none]
resolved_color	return location for the resolved color.	[out]

Returns

`TRUE` if the color has been resolved

Since: 3.0

---

gtk_symbolic_color_to_string ()

```
char *
gtk_symbolic_color_to_string (GtkSymbolicColor *color);
```

`gtk_symbolic_color_to_string` has been deprecated since version 3.8 and should not be used in newly-written code.

GtkSymbolicColor is deprecated.

Converts the given `color` to a string representation. This is useful both for debugging and for serialization of strings. The format of the string may change between different versions of GTK, but it is guaranteed that the GTK css parser is able to read the string and create the same symbolic color from it.

Parameters

color	color to convert to a string

Returns

a new string representing `color`

Types and Values

GtkSymbolicColor

```
typedef struct _GtkSymbolicColor GtkSymbolicColor;
```

GtkGradient

GtkGradient — Gradients

Functions

GtkGradient *	gtk_gradient_new_linear ()
GtkGradient *	gtk_gradient_new_radial ()
void	gtk_gradient_add_color_stop ()
GtkGradient *	gtk_gradient_ref ()
void	gtk_gradient_unref ()
gboolean	gtk_gradient_resolve ()
cairo_pattern_t *	gtk_gradient_resolve_for_context ()
char *	gtk_gradient_to_string ()

Types and Values

GtkGradient

Includes

```
#include <gtk/gtk.h>
```

Description

GtkGradient is a boxed type that represents a gradient. It is the result of parsing a gradient expression. To obtain the gradient represented by a GtkGradient, it has to be resolved with `gtk_gradient_resolve()`, which replaces all symbolic color references by the colors they refer to (in a given context) and constructs a cairo_pattern_t value.

It is not normally necessary to deal directly with GtkGradients, since they are mostly used behind the scenes by GtkStyleContext and GtkCssProvider.

GtkGradient is deprecated. It was used internally by GTK's CSS engine to represent gradients. As its handling is not conforming to modern web standards, it is not used anymore. If you want to use gradients in your own code, please use Cairo directly.

Functions

gtk_gradient_new_linear ()

```
GtkGradient *
gtk_gradient_new_linear (gdouble x0,
                         gdouble y0,
                         gdouble x1,
                         gdouble y1);
```

`gtk_gradient_new_linear` has been deprecated since version 3.8 and should not be used in newly-written code.

GtkGradient is deprecated.

Creates a new linear gradient along the line defined by (x0, y0) and (x1, y1). Before using the gradient a number of stop colors must be added through `gtk_gradient_add_color_stop()`.

Parameters

x0	X coordinate of the starting point
y0	Y coordinate of the starting point
x1	X coordinate of the end point
y1	Y coordinate of the end point

Returns

A newly created GtkGradient

Since: 3.0

---

gtk_gradient_new_radial ()

```
GtkGradient *
gtk_gradient_new_radial (gdouble x0,
                         gdouble y0,
                         gdouble radius0,
                         gdouble x1,
                         gdouble y1,
                         gdouble radius1);
```

`gtk_gradient_new_radial` has been deprecated since version 3.8 and should not be used in newly-written code.

GtkGradient is deprecated.

Creates a new radial gradient along the two circles defined by (x0, y0, radius0) and (x1, y1, radius1). Before using the gradient a number of stop colors must be added through `gtk_gradient_add_color_stop()`.

Parameters

x0	X coordinate of the start circle
y0	Y coordinate of the start circle
radius0	radius of the start circle
x1	X coordinate of the end circle
y1	Y coordinate of the end circle
radius1	radius of the end circle

Returns

A newly created GtkGradient

Since: 3.0

---

gtk_gradient_add_color_stop ()

```
void
gtk_gradient_add_color_stop (GtkGradient *gradient,
                             gdouble offset,
                             GtkSymbolicColor *color);
```

`gtk_gradient_add_color_stop` has been deprecated since version 3.8 and should not be used in newly-written code.

GtkGradient is deprecated.

Adds a stop color to `gradient` .

Parameters

gradient	a GtkGradient
offset	offset for the color stop
color	color to use

Since: 3.0

---

gtk_gradient_ref ()

```
GtkGradient *
gtk_gradient_ref (GtkGradient *gradient);
```

`gtk_gradient_ref` has been deprecated since version 3.8 and should not be used in newly-written code.

GtkGradient is deprecated.

Increases the reference count of `gradient` .

Parameters

gradient	a GtkGradient

Returns

The same `gradient`

Since: 3.0

---

gtk_gradient_unref ()

```
void
gtk_gradient_unref (GtkGradient *gradient);
```

`gtk_gradient_unref` has been deprecated since version 3.8 and should not be used in newly-written code.

GtkGradient is deprecated.

Decreases the reference count of `gradient` , freeing its memory if the reference count reaches 0.

Parameters

gradient	a GtkGradient	

Since: 3.0

---

gtk_gradient_resolve ()

```
gboolean
gtk_gradient_resolve (GtkGradient *gradient,
                      GtkStyleProperties *props,
                      cairo_pattern_t **resolved_gradient);
```

`gtk_gradient_resolve` has been deprecated since version 3.8 and should not be used in newly-written code.

GtkGradient is deprecated.

If `gradient` is resolvable, `resolved_gradient` will be filled in with the resolved gradient as a cairo_pattern_t, and `TRUE` will be returned. Generally, if `gradient` can’t be resolved, it is due to it being defined on top of a named color that doesn't exist in `props` .

Parameters

gradient	a GtkGradient	
props	GtkStyleProperties to use when resolving named colors	
resolved_gradient	return location for the resolved pattern.	[out]

Returns

`TRUE` if the gradient has been resolved

Since: 3.0

---

gtk_gradient_resolve_for_context ()

```
cairo_pattern_t *
gtk_gradient_resolve_for_context (GtkGradient *gradient,
                                  GtkStyleContext *context);
```

`gtk_gradient_resolve_for_context` is deprecated and should not be used in newly-written code.

---

gtk_gradient_to_string ()

```
char *
gtk_gradient_to_string (GtkGradient *gradient);
```

`gtk_gradient_to_string` has been deprecated since version 3.8 and should not be used in newly-written code.

GtkGradient is deprecated.

Creates a string representation for `gradient` that is suitable for using in GTK CSS files.

Parameters

gradient	the gradient to print

Returns

A string representation for `gradient`

Types and Values

GtkGradient

```
typedef struct _GtkGradient GtkGradient;
```

Resource Files

Resource Files — Deprecated routines for handling resource files

Functions

GScanner *	gtk_rc_scanner_new ()
GtkStyle *	gtk_rc_get_style ()
GtkStyle *	gtk_rc_get_style_by_paths ()
void	gtk_rc_parse ()
void	gtk_rc_parse_string ()
gboolean	gtk_rc_reparse_all ()

gboolean	gtk_rc_reparse_all_for_settings ()
void	gtk_rc_reset_styles ()
void	gtk_rc_add_default_file ()
gchar **	gtk_rc_get_default_files ()
void	gtk_rc_set_default_files ()
guint	gtk_rc_parse_color ()
guint	gtk_rc_parse_color_full ()
guint	gtk_rc_parse_state ()
guint	gtk_rc_parse_priority ()
gchar *	gtk_rc_find_module_in_path ()
gchar *	gtk_rc_find_pixmap_in_path ()
gchar *	gtk_rc_get_module_dir ()
gchar *	gtk_rc_get_im_module_path ()
gchar *	gtk_rc_get_im_module_file ()
gchar *	gtk_rc_get_theme_dir ()
GtkRcStyle *	gtk_rc_style_new ()
GtkRcStyle *	gtk_rc_style_copy ()

Types and Values

	GtkRcStyle
struct	GtkRcStyleClass
enum	GtkRcFlags
enum	GtkRcTokenType
enum	GtkPathPriorityType
enum	GtkPathType

Object Hierarchy

```
GObject
└── GtkRcStyle
```

Includes

```
#include <gtk/gtk.h>
```

Description

GTK+ provides resource file mechanism for configuring various aspects of the operation of a GTK+ program at runtime.

> In GTK+ 3.0, resource files have been deprecated and replaced by CSS-like style sheets, which are understood by GtkCssProvider.

Default Files

An application can cause GTK+ to parse a specific RC file by calling `gtk_rc_parse()`. In addition to this, certain files will be read at the end of `gtk_init()`. Unless modified, the files looked for will be

SYSCONFDIR/gtk-2.0/gtkrc and .gtkrc-3.0 in the users home directory. (SYSCONFDIR defaults to /usr/local/etc. It can be changed with the --prefix or --sysconfdir options when configuring GTK+.)

The set of these “default” files can be retrieved with gtk_rc_get_default_files() and modified with gtk_rc_add_default_file() and gtk_rc_set_default_files(). Additionally, the GTK2_RC_FILES environment variable can be set to a G_SEARCHPATH_SEPARATOR_S-separated list of files in order to overwrite the set of default files at runtime.

Locale Specific Files

For each RC file, in addition to the file itself, GTK+ will look for a locale-specific file that will be parsed after the main file. For instance, if LANG is set to ja_JP.ujis, when loading the default file ~/.gtkrc then GTK+ looks for ~/.gtkrc.ja_JP and ~/.gtkrc.ja, and parses the first of those that exists.

Pathnames and Patterns

A resource file defines a number of styles and key bindings and attaches them to particular widgets. The attachment is done by the widget, widget_class, and class declarations. As an example of such a statement:

```
1                                                   widget "mywindow.*.GtkEntry" style "my-entry-class"
```

attaches the style "my-entry-class" to all widgets whose “widget path” matches the “pattern” "mywindow.*.GtkEntry". That is, all GtkEntry widgets which are part of a GtkWindow named "mywindow".

The patterns here are given in the standard shell glob syntax. The "?" wildcard matches any character, while "*" matches zero or more of any character. The three types of matching are against the widget path, the “class path” and the class hierarchy. Both the widget path and the class path consist of a "." separated list of all the parents of the widget and the widget itself from outermost to innermost. The difference is that in the widget path, the name assigned by gtk_widget_set_name() is used if present, otherwise the class name of the widget, while for the class path, the class name is always used.

Since GTK+ 2.10, widget_class paths can also contain <classname> substrings, which are matching the class with the given name and any derived classes. For instance,

```
1                                                   widget_class "*<GtkMenuItem>.GtkLabel" style "my-style"
```

will match GtkLabel widgets which are contained in any kind of menu item.

So, if you have a GtkEntry named "myentry", inside of a horizontal box in a window named "mywindow", then the widget path is: "mywindow.GtkHBox.myentry" while the class path is: "GtkWindow.GtkHBox.GtkEntry".

Matching against class is a little different. The pattern match is done against all class names in the widgets class hierarchy (not the layout hierarchy) in sequence, so the pattern:

```
1                                                   class "GtkButton" style "my-style"
```

will match not just GtkButton widgets, but also GtkToggleButton and GtkCheckButton widgets, since those classes derive from GtkButton.

Additionally, a priority can be specified for each pattern, and styles override other styles first by priority, then by pattern type and then by order of specification (later overrides earlier). The priorities that can be specified are (highest to lowest):

- highest
- rc

- `theme`
- `application`
- `gtk`
- `lowest`

`rc` is the default for styles read from an RC file, `theme` is the default for styles read from theme RC files, `application` should be used for styles an application sets up, and `gtk` is used for styles that GTK+ creates internally.

Theme gtkrc Files

Theme RC files are loaded first from under the `~/.themes/`, then from the directory from `gtk_rc_get_theme_dir()`. The files looked at will be `gtk-3.0/gtkrc`.

When the application prefers dark themes (see the "gtk-application-prefer-dark-theme" property for details), `gtk-3.0/gtkrc-dark` will be loaded first, and if not present `gtk-3.0/gtkrc` will be loaded.

Optimizing RC Style Matches

Everytime a widget is created and added to the layout hierarchy of a GtkWindow ("anchored" to be exact), a list of matching RC styles out of all RC styles read in so far is composed. For this, every RC style is matched against the widgets class path, the widgets name path and widgets inheritance hierarchy. As a consequence, significant slowdown can be caused by utilization of many RC styles and by using RC style patterns that are slow or complicated to match against a given widget. The following ordered list provides a number of advices (prioritized by effectiveness) to reduce the performance overhead associated with RC style matches:

1. Move RC styles for specific applications into RC files dedicated to those applications and parse application specific RC files only from applications that are affected by them. This reduces the overall amount of RC styles that have to be considered for a match across a group of applications.
2. Merge multiple styles which use the same matching rule, for instance:

   is faster to match as:
3. Use of wildcards should be avoided, this can reduce the individual RC style match to a single integer comparison in most cases.
4. To avoid complex recursive matching, specification of full class names (for `class` matches) or full path names (for `widget` and `widget_class` matches) is to be preferred over shortened names containing `"*"` or `"?"`.
5. If at all necessary, wildcards should only be used at the tail or head of a pattern. This reduces the match complexity to a string comparison per RC style.
6. When using wildcards, use of `"?"` should be preferred over `"*"`. This can reduce the matching complexity from O(n^2) to O(n). For example `"Gtk*Box"` can be turned into `"Gtk?Box"` and will still match GtkHBox and GtkVBox.
7. The use of `"*"` wildcards should be restricted as much as possible, because matching `"A*B*C*RestString"` can result in matching complexities of O(n^2) worst case.

Toplevel Declarations

An RC file is a text file which is composed of a sequence of declarations. "#" characters delimit comments and the portion of a line after a "#" is ignored when parsing an RC file.

The possible toplevel declarations are:

- `binding name  { ... }`

  Declares a binding set.

- `class pattern  [ style | binding ][ : priority ]  name`

  Specifies a style or binding set for a particular branch of the inheritance hierarchy.

- `include filename`

  Parses another file at this point. If filename is not an absolute filename, it is searched in the directories of the currently open RC files.

  GTK+ also tries to load a locale-specific variant of the included file.

- `module_path path`

  Sets a path (a list of directories separated by colons) that will be searched for theme engines referenced in RC files.

- `pixmap_path path`

  Sets a path (a list of directories separated by colons) that will be searched for pixmaps referenced in RC files.

- `im_module_file pathname`

  Sets the pathname for the IM modules file. Setting this from RC files is deprecated; you should use the environment variable `GTK_IM_MODULE_FILE` instead.

- `style name [ = parent ] { ... }`

  Declares a style.

- `widget pattern  [ style | binding ][ : priority ]  name`

  Specifies a style or binding set for a particular group of widgets by matching on the widget pathname.

- `widget_class pattern  [ style | binding ][ : priority ]  name`

  Specifies a style or binding set for a particular group of widgets by matching on the class pathname.

- setting = value

  Specifies a value for a setting. Note that settings in RC files are overwritten by system-wide settings (which are managed by an XSettings manager on X11).

Styles

A RC style is specified by a `style` declaration in a RC file, and then bound to widgets with a `widget`, `widget_class`, or `class` declaration. All styles applying to a particular widget are composited together with

`widget` declarations overriding `widget_class` declarations which, in turn, override `class` declarations. Within each type of declaration, later declarations override earlier ones.

Within a `style` declaration, the possible elements are:

- `bg[state] = color`

  Sets the color used for the background of most widgets.

- `fg[state] = color`

  Sets the color used for the foreground of most widgets.

- `base[state] = color`

  Sets the color used for the background of widgets displaying editable text. This color is used for the background of, among others, GtkTextView, GtkEntry.

- `text[state] =  color`

  Sets the color used for foreground of widgets using `base` for the background color.

- `xthickness =  number`

  Sets the xthickness, which is used for various horizontal padding values in GTK+.

- `ythickness =  number`

  Sets the ythickness, which is used for various vertical padding values in GTK+.

- `bg_pixmap[state] =  pixmap`

  Sets a background pixmap to be used in place of the `bg` color (or for GtkText, in place of the `base` color. The special value `"<parent>"` may be used to indicate that the widget should use the same background pixmap as its parent. The special value `"<none>"` may be used to indicate no background pixmap.

- `font = font`

  Starting with GTK+ 2.0, the “font” and “fontset” declarations are ignored; use “font_name” declarations instead.

- `fontset = font`

  Starting with GTK+ 2.0, the “font” and “fontset” declarations are ignored; use “font_name” declarations instead.

- `font_name = font`

  Sets the font for a widget. font must be a Pango font name, e.g. “Sans Italic 10” . For details about Pango font names, see `pango_font_description_from_string()`.

- `stock["stock-id"] = { icon source specifications }`

  Defines the icon for a stock item.

- `color["color-name"] = color specification`

  Since 2.10, this element can be used to defines symbolic colors. See below for the syntax of color specifications.

- `engine "engine" { engine-specific settings }`

  Defines the engine to be used when drawing with this style.

- `class::property = value`

  Sets a style property for a widget class.

The colors and background pixmaps are specified as a function of the state of the widget. The states are:

- `NORMAL`

  A color used for a widget in its normal state.

- `ACTIVE`

  A variant of the `NORMAL` color used when the widget is in the `GTK_STATE_ACTIVE` state, and also for the trough of a ScrollBar, tabs of a NoteBook other than the current tab and similar areas. Frequently, this should be a darker variant of the `NORMAL` color.

- `PRELIGHT`

  A color used for widgets in the `GTK_STATE_PRELIGHT` state. This state is the used for Buttons and MenuItems that have the mouse cursor over them, and for their children.

- `SELECTED`

  A color used to highlight data selected by the user. for instance, the selected items in a list widget, and the selection in an editable widget.

- `INSENSITIVE`

  A color used for the background of widgets that have been set insensitive with `gtk_widget_set_sensitive()`.

Color Format

Colors can be specified as a string containing a color name (GTK+ knows all names from the X color database `/usr/lib/X11/rgb.txt`), in one of the hexadecimal forms `#rrrrggggbbbb`, `#rrrgggbbb`, `#rrggbb`, or `#rgb`, where `r`, `g` and `b` are hex digits, or they can be specified as a triplet `{ r, g, b}`, where `r`, `g` and `b` are either integers in the range 0-65535 or floats in the range 0.0-1.0.

Since 2.10, colors can also be specified by refering to a symbolic color, as follows: `@color-name`, or by using expressions to combine colors. The following expressions are currently supported:

- mix (factor, color1, color2)

  Computes a new color by mixing color1 and color2. The factor determines how close the new color is to color1. A factor of 1.0 gives pure color1, a factor of 0.0 gives pure color2.

- shade (factor, color)

  Computes a lighter or darker variant of color. A factor of 1.0 leaves the color unchanged, smaller factors yield darker colors, larger factors yield lighter colors.

- lighter (color)

  This is an abbreviation for `shade (1.3, color)`.

- darker (color)

  This is an abbreviation for `shade (0.7, color)`.

Here are some examples of color expressions:

```
1                                          mix (0.5, "red", "blue")
2                                          shade (1.5, mix (0.3, "#0abbc0", { 0.3, 0.5,
```

```
3                                                   0.9 }))
                                                    lighter (@foreground)
```

In a `stock` definition, icon sources are specified as a 4-tuple of image filename or icon name, text direction, widget state, and size, in that order. Each icon source specifies an image filename or icon name to use with a given direction, state, and size. Filenames are specified as a string such as `"itemltr.png"`, while icon names (looked up in the current icon theme), are specified with a leading `@`, such as `@"item-ltr"`. The `*` character can be used as a wildcard, and if direction/state/size are omitted they default to `*`. So for example, the following specifies different icons to use for left-to-right and right-to-left languages:

```
1                                                   stock["my-stock-item"] =
2                                                   {
3                                                     { "itemltr.png", LTR, *, * },
4                                                     { "itemrtl.png", RTL, *, * }
5                                                   }
```

This could be abbreviated as follows:

```
1                                                   stock["my-stock-item"] =
2                                                   {
3                                                     { "itemltr.png", LTR },
4                                                     { "itemrtl.png", RTL }
5                                                   }
```

You can specify custom icons for specific sizes, as follows:

```
1                                                   stock["my-stock-item"] =
2                                                   {
3                                                     { "itemmenusize.png", *, *, "gtk-menu" },
4                                                     { "itemtoolbarsize.png", *, *, "gtk-large-
5                                                   toolbar" }
6                                                     { "itemgeneric.png" } // implicit *, *, *
                                                    as a fallback
                                                    }
```

The sizes that come with GTK+ itself are `"gtk-menu"`, `"gtk-small-toolbar"`, `"gtk-large-toolbar"`, `"gtk-button"`, `"gtk-dialog"`. Applications can define other sizes.

It’s also possible to use custom icons for a given state, for example:

```
1                                                   stock["my-stock-item"] =
2                                                   {
3                                                     { "itemprelight.png", *, PRELIGHT },
4                                                     { "iteminsensitive.png", *, INSENSITIVE },
5                                                     { "itemgeneric.png" } // implicit *, *, *
6                                                   as a fallback
                                                    }
```

When selecting an icon source to use, GTK+ will consider text direction most important, state second, and size third. It will select the best match based on those criteria. If an attribute matches exactly (e.g. you specified `PRELIGHT` or specified the size), GTK+ won’t modify the image; if the attribute matches with a wildcard, GTK+ will scale or modify the image to match the state and size the user requested.

Key bindings

Key bindings allow the user to specify actions to be taken on particular key presses. The form of a binding set declaration is:

```
1                                                   binding name {
2                                                     bind key {
3                                                       signalname (param, ...)
4                                                       ...
5                                                     }
6                                                     ...
7                                                   }
```

`key` is a string consisting of a series of modifiers followed by the name of a key. The modifiers can be:

- <alt>
- <ctl>
- <control>
- <meta>
- <hyper>
- <super>
- <mod1>
- <mod2>
- <mod3>
- <mod4>
- <mod5>
- <release>
- <shft>
- <shift>

<shft> is an alias for <shift>, <ctl> is an alias for <control>, and <alt> is an alias for <mod1>.

The action that is bound to the key is a sequence of signal names (strings) followed by parameters for each signal. The signals must be action signals. (See g_signal_new()). Each parameter can be a float, integer, string, or unquoted string representing an enumeration value. The types of the parameters specified must match the types of the parameters of the signal.

Binding sets are connected to widgets in the same manner as styles, with one difference: Binding sets override other binding sets first by pattern type, then by priority and then by order of specification. The priorities that can be specified and their default values are the same as for styles.

Functions

gtk_rc_scanner_new ()

```
GScanner *
gtk_rc_scanner_new (void);
```

gtk_rc_scanner_new has been deprecated since version 3.0 and should not be used in newly-written code.

Use GtkCssProvider instead

[skip]

---

gtk_rc_get_style ()

```
GtkStyle *
gtk_rc_get_style (GtkWidget *widget);
```

gtk_rc_get_style has been deprecated since version 3.0 and should not be used in newly-written code.

Use GtkStyleContext instead

Finds all matching RC styles for a given widget, composites them together, and then creates a GtkStyle representing the composite appearance. (GTK+ actually keeps a cache of previously created styles, so a new style may not be created.)

Parameters

widget	a GtkWidget	

Returns

the resulting style. No refcount is added to the returned style, so if you want to save this style around, you should add a reference yourself.

[transfer none]

---

gtk_rc_get_style_by_paths ()

```
GtkStyle *
gtk_rc_get_style_by_paths (GtkSettings *settings,
                           const char *widget_path,
                           const char *class_path,
                           GType type);
```

`gtk_rc_get_style_by_paths` has been deprecated since version 3.0 and should not be used in newly-written code.

Use GtkStyleContext instead

Creates up a GtkStyle from styles defined in a RC file by providing the raw components used in matching. This function may be useful when creating pseudo-widgets that should be themed like widgets but don't actually have corresponding GTK+ widgets. An example of this would be items inside a GNOME canvas widget.

The action of `gtk_rc_get_style()` is similar to:

```
1                                       gtk_widget_path (widget, NULL, &path, NULL);
2                                       gtk_widget_class_path (widget, NULL,
3                                       &class_path, NULL);
4                                       gtk_rc_get_style_by_paths
5                                       (gtk_widget_get_settings (widget),
                                                                  path, class_path,
                                                                  G_OBJECT_TYPE
                                        (widget));
```

Parameters

settings	a GtkSettings object	
widget_path	the widget path to use when looking up the style, or `NULL` if no matching against the widget path should be done.	[allow-none]
class_path	the class path to use when looking up the style, or `NULL` if no matching against the class path should be	[allow-none]

	done.
type	a type that will be used along with parent types of this type when matching against class styles, or G_TYPE_NONE

Returns

A style created by matching with the supplied paths, or `NULL` if nothing matching was specified and the default style should be used. The returned value is owned by GTK+ as part of an internal cache, so you must call `g_object_ref()` on the returned value if you want to keep a reference to it.

[nullable][transfer none]

---

gtk_rc_parse ()

```
void
gtk_rc_parse (const gchar *filename);
```

`gtk_rc_parse` has been deprecated since version 3.0 and should not be used in newly-written code.

Use GtkCssProvider instead.

Parses a given resource file.

Parameters

filename	the filename of a file to parse. If `filename` is not absolute, it is searched in the current directory.

---

gtk_rc_parse_string ()

```
void
gtk_rc_parse_string (const gchar *rc_string);
```

`gtk_rc_parse_string` has been deprecated since version 3.0 and should not be used in newly-written code.

Use GtkCssProvider instead.

Parses resource information directly from a string.

Parameters

rc_string	a string to parse.

---

gtk_rc_reparse_all ()

```
gboolean
gtk_rc_reparse_all (void);
```

`gtk_rc_reparse_all` has been deprecated since version 3.0 and should not be used in newly-written code.

Use GtkCssProvider instead.

If the modification time on any previously read file for the default GtkSettings has changed, discard all style information and then reread all previously read RC files.

Returns

`TRUE` if the files were reread.

---

gtk_rc_reparse_all_for_settings ()

```
gboolean
gtk_rc_reparse_all_for_settings (GtkSettings *settings,
                                 gboolean force_load);
```

`gtk_rc_reparse_all_for_settings` has been deprecated since version 3.0 and should not be used in newly-written code.

Use GtkCssProvider instead.

If the modification time on any previously read file for the given GtkSettings has changed, discard all style information and then reread all previously read RC files.

Parameters

settings	a GtkSettings
force_load	load whether or not anything changed

Returns

`TRUE` if the files were reread.

---

gtk_rc_reset_styles ()

```
void
gtk_rc_reset_styles (GtkSettings *settings);
```

`gtk_rc_reset_styles` has been deprecated since version 3.0 and should not be used in newly-written code.

Use GtkCssProvider instead.

This function recomputes the styles for all widgets that use a particular GtkSettings object. (There is one GtkSettings object per GdkScreen, see `gtk_settings_get_for_screen()`); It is useful when some global parameter has changed that affects the appearance of all widgets, because when a widget gets a new style, it will

both redraw and recompute any cached information about its appearance. As an example, it is used when the default font size set by the operating system changes. Note that this function doesn't affect widgets that have a style set explicitly on them with `gtk_widget_set_style()`.

Parameters

settings	a GtkSettings

Since: 2.4

---

gtk_rc_add_default_file ()

```
void
gtk_rc_add_default_file (const gchar *filename);
```

`gtk_rc_add_default_file` has been deprecated since version 3.0 and should not be used in newly-written code.

Use GtkStyleContext with a custom GtkStyleProvider instead

Adds a file to the list of files to be parsed at the end of `gtk_init()`.

Parameters

filename	the pathname to the file. If `filename` is not absolute, it is searched in the current directory.	[type filename]

---

gtk_rc_get_default_files ()

```
gchar **
gtk_rc_get_default_files (void);
```

`gtk_rc_get_default_files` has been deprecated since version 3.0 and should not be used in newly-written code.

Use GtkStyleContext instead

Retrieves the current list of RC files that will be parsed at the end of `gtk_init()`.

Returns

A `NULL`-terminated array of filenames. This memory is owned by GTK+ and must not be freed by the application. If you want to store this information, you should make a copy.

[transfer none][array zero-terminated=1][element-type filename]

---

gtk_rc_set_default_files ()

```
void
gtk_rc_set_default_files (gchar **filenames);
```

`gtk_rc_set_default_files` has been deprecated since version 3.0 and should not be used in newly-written code.

Use GtkStyleContext with a custom GtkStyleProvider instead

Sets the list of files that GTK+ will read at the end of `gtk_init()`.

Parameters

filenames	A `NULL`-terminated list of filenames.	[array zero-terminated=1][element-type filename]

---

gtk_rc_parse_color ()

```
guint
gtk_rc_parse_color (GScanner *scanner,
                    GdkColor *color);
```

`gtk_rc_parse_color` has been deprecated since version 3.0 and should not be used in newly-written code.

Use GtkCssProvider instead

Parses a color in the format expected in a RC file.

Note that theme engines should use `gtk_rc_parse_color_full()` in order to support symbolic colors.

Parameters

scanner	a GScanner	
color	a pointer to a GdkColor in which to store the result.	[out]

Returns

`G_TOKEN_NONE` if parsing succeeded, otherwise the token that was expected but not found

---

gtk_rc_parse_color_full ()

```
guint
gtk_rc_parse_color_full (GScanner *scanner,
                         GtkRcStyle *style,
                         GdkColor *color);
```

`gtk_rc_parse_color_full` has been deprecated since version 3.0 and should not be used in newly-written code.

Use GtkCssProvider instead

Parses a color in the format expected in a RC file. If `style` is not `NULL`, it will be consulted to resolve references to symbolic colors.

Parameters

scanner	a GScanner	
style	a GtkRcStyle, or `NULL`.	[allow-none]
color	a pointer to a GdkColor in which to store the result.	[out]

Returns

`G_TOKEN_NONE` if parsing succeeded, otherwise the token that was expected but not found

Since: 2.12

---

gtk_rc_parse_state ()

```
guint
gtk_rc_parse_state (GScanner *scanner,
                    GtkStateType *state);
```

`gtk_rc_parse_state` has been deprecated since version 3.0 and should not be used in newly-written code.

Use GtkCssProvider instead

Parses a GtkStateType variable from the format expected in a RC file.

Parameters

scanner	a GScanner (must be initialized for parsing an RC file)	
state	A pointer to a GtkStateType variable in which to store the result.	[out]

Returns

`G_TOKEN_NONE` if parsing succeeded, otherwise the token that was expected but not found.

---

gtk_rc_parse_priority ()

```
guint
gtk_rc_parse_priority (GScanner *scanner,
                       GtkPathPriorityType *priority);
```

`gtk_rc_parse_priority` has been deprecated since version 3.0 and should not be used in newly-written code.

Use GtkCssProvider instead

Parses a GtkPathPriorityType variable from the format expected in a RC file.

Parameters

scanner	a GScanner (must be initialized for parsing an RC file)
priority	A pointer to GtkPathPriorityType variable in which to store the result.

Returns

`G_TOKEN_NONE` if parsing succeeded, otherwise the token that was expected but not found.

---

gtk_rc_find_module_in_path ()

```
gchar *
gtk_rc_find_module_in_path (const gchar *module_file);
```

`gtk_rc_find_module_in_path` has been deprecated since version 3.0 and should not be used in newly-written code.

Use GtkCssProvider instead.

Searches for a theme engine in the GTK+ search path. This function is not useful for applications and should not be used.

Parameters

module_file	name of a theme engine

Returns

The filename, if found (must be freed with `g_free()`), otherwise `NULL`.

[type filename]

---

gtk_rc_find_pixmap_in_path ()

```
gchar *
gtk_rc_find_pixmap_in_path (GtkSettings *settings,
                            GScanner *scanner,
                            const gchar *pixmap_file);
```

`gtk_rc_find_pixmap_in_path` has been deprecated since version 3.0 and should not be used in newly-written code.

Use GtkCssProvider instead.

Looks up a file in pixmap path for the specified GtkSettings. If the file is not found, it outputs a warning message using `g_warning()` and returns `NULL`.

Parameters

settings	a GtkSettings
scanner	Scanner used to get line number information for the warning message, or NULL
pixmap_file	name of the pixmap file to locate.

Returns

the filename.

[type filename]

---

gtk_rc_get_module_dir ()

```
gchar *
gtk_rc_get_module_dir (void);
```

`gtk_rc_get_module_dir` has been deprecated since version 3.0 and should not be used in newly-written code.

Use GtkCssProvider instead.

Returns a directory in which GTK+ looks for theme engines. For full information about the search for theme engines, see the docs for `GTK_PATH` in Running GTK+ Applications.

Returns

the directory. (Must be freed with `g_free()`).

[type filename]

---

gtk_rc_get_im_module_path ()

```
gchar *
gtk_rc_get_im_module_path (void);
```

`gtk_rc_get_im_module_path` has been deprecated since version 3.0 and should not be used in newly-written code.

Use GtkCssProvider instead.

Obtains the path in which to look for IM modules. See the documentation of the `GTK_PATH` environment variable for more details about looking up modules. This function is useful solely for utilities supplied with GTK+ and should not be used by applications under normal circumstances.

Returns

a newly-allocated string containing the path in which to look for IM modules.

[type filename]

---

gtk_rc_get_im_module_file ()

```
gchar *
gtk_rc_get_im_module_file (void);
```

`gtk_rc_get_im_module_file` has been deprecated since version 3.0 and should not be used in newly-written code.

Use GtkCssProvider instead.

Obtains the path to the IM modules file. See the documentation of the `GTK_IM_MODULE_FILE` environment variable for more details.

Returns

a newly-allocated string containing the name of the file listing the IM modules available for loading.

[type filename]

---

gtk_rc_get_theme_dir ()

```
gchar *
gtk_rc_get_theme_dir (void);
```

`gtk_rc_get_theme_dir` has been deprecated since version 3.0 and should not be used in newly-written code.

Use GtkCssProvider instead.

Returns the standard directory in which themes should be installed. (GTK+ does not actually use this directory itself.)

Returns

The directory (must be freed with `g_free()`).

---

gtk_rc_style_new ()

```
GtkRcStyle *
gtk_rc_style_new (void);
```

`gtk_rc_style_new` has been deprecated since version 3.0 and should not be used in newly-written code.

Use GtkCssProvider instead.

Creates a new GtkRcStyle with no fields set and a reference count of 1.

Returns

the newly-created GtkRcStyle

---

gtk_rc_style_copy ()

```
GtkRcStyle *
gtk_rc_style_copy (GtkRcStyle *orig);
```

`gtk_rc_style_copy` has been deprecated since version 3.0 and should not be used in newly-written code.

Use GtkCssProvider instead.

Makes a copy of the specified GtkRcStyle. This function will correctly copy an RC style that is a member of a class derived from GtkRcStyle.

Parameters

orig	the style to copy

Returns

the resulting GtkRcStyle.

[transfer full]

Types and Values

GtkRcStyle

```
typedef struct {
  gchar *name;
  gchar *bg_pixmap_name[5];
  PangoFontDescription *font_desc;

  GtkRcFlags color_flags[5];
  GdkColor   fg[5];
  GdkColor   bg[5];
  GdkColor   text[5];
  GdkColor   base[5];

  gint xthickness;
  gint ythickness;
} GtkRcStyle;
```

The GtkRcStyle is used to represent a set of information about the appearance of a widget. This can later be composited together with other GtkRcStyle<!-- -->s to form a GtkStyle.

Members

gchar *`name`;	Name
gchar *`bg_pixmap_name`[5];	Pixmap name
PangoFontDescription *`font_desc`;	A PangoFontDescription
GtkRcFlags `color_flags`[5];	GtkRcFlags
GdkColor `fg`[5];	Foreground colors

GdkColor `bg`[5];	Background colors
GdkColor `text`[5];	Text colors
GdkColor `base`[5];	Base colors
gint `xthickness`;	X thickness
gint `ythickness`;	Y thickness

---

struct GtkRcStyleClass

```
struct GtkRcStyleClass {
  GObjectClass parent_class;


  /* Create an empty RC style of the same type as this RC style.
   * The default implementation, which does
   * g_object_new (G_OBJECT_TYPE (style), NULL);
   * should work in most cases.
   */
  GtkRcStyle * (*create_rc_style) (GtkRcStyle *rc_style);

  /* Fill in engine specific parts of GtkRcStyle by parsing contents
   * of brackets. Returns G_TOKEN_NONE if successful, otherwise returns
   * the token it expected but didn't get.
   */
  guint     (*parse)  (GtkRcStyle   *rc_style,
                       GtkSettings  *settings,
                       GScanner     *scanner);

  /* Combine RC style data from src into dest. If overridden, this
   * function should chain to the parent.
   */
  void      (*merge)  (GtkRcStyle *dest,
                       GtkRcStyle *src);

  /* Create an empty style suitable to this RC style
   */
  GtkStyle * (*create_style) (GtkRcStyle *rc_style);
};
```

Members

`create_rc_style` ()

`parse` ()

`merge` ()

`create_style` ()

---

enum GtkRcFlags

Deprecated

Members

GTK_RC_FG	Deprecated
GTK_RC_BG	Deprecated
GTK_RC_TEXT	Deprecated
GTK_RC_BASE	Deprecated

---

enum GtkRcTokenType

`GtkRcTokenType` has been deprecated since version 3.0 and should not be used in newly-written code.

Use GtkCssProvider instead.

The GtkRcTokenType enumeration represents the tokens in the RC file. It is exposed so that theme engines can reuse these tokens when parsing the theme-engine specific portions of a RC file.

Members

GTK_RC_TOKEN_INVALID	Deprecated
GTK_RC_TOKEN_INCLUDE	Deprecated
GTK_RC_TOKEN_NORMAL	Deprecated
GTK_RC_TOKEN_ACTIVE	Deprecated
GTK_RC_TOKEN_PRELIGHT	Deprecated
GTK_RC_TOKEN_SELECTED	Deprecated
GTK_RC_TOKEN_INSENSITIVE	Deprecated
GTK_RC_TOKEN_FG	Deprecated
GTK_RC_TOKEN_BG	Deprecated
GTK_RC_TOKEN_TEXT	Deprecated
GTK_RC_TOKEN_BASE	Deprecated
GTK_RC_TOKEN_XTHICKNESS	Deprecated
GTK_RC_TOKEN_YTHICKNESS	Deprecated
GTK_RC_TOKEN_FONT	Deprecated
GTK_RC_TOKEN_FONTSET	Deprecated
GTK_RC_TOKEN_FONT_NAME	Deprecated
GTK_RC_TOKEN_BG_PIXMAP	Deprecated
GTK_RC_TOKEN_PIXMAP_PATH	Deprecated
GTK_RC_TOKEN_STYLE	Deprecated
GTK_RC_TOKEN_BINDING	Deprecated
GTK_RC_TOKEN_BIND	Deprecated
GTK_RC_TOKEN_WIDGET	Deprecated
GTK_RC_TOKEN_WIDGET_CLASS	Deprecated
GTK_RC_TOKEN_CLASS	Deprecated
GTK_RC_TOKEN_LOWEST	Deprecated
GTK_RC_TOKEN_GTK	Deprecated
GTK_RC_TOKEN_APPLICATION	Deprecated
GTK_RC_TOKEN_THEME	Deprecated
GTK_RC_TOKEN_RC	Deprecated

GTK_RC_TOKEN_HIGHEST	Deprecated
GTK_RC_TOKEN_ENGINE	Deprecated
GTK_RC_TOKEN_MODULE_PATH	Deprecated
GTK_RC_TOKEN_IM_MODULE_PATH	Deprecated
GTK_RC_TOKEN_IM_MODULE_FILE	Deprecated
GTK_RC_TOKEN_STOCK	Deprecated
GTK_RC_TOKEN_LTR	Deprecated
GTK_RC_TOKEN_RTL	Deprecated
GTK_RC_TOKEN_COLOR	Deprecated
GTK_RC_TOKEN_UNBIND	Deprecated
GTK_RC_TOKEN_LAST	Deprecated

---

enum GtkPathPriorityType

`GtkPathPriorityType` has been deprecated since version 3.0 and should not be used in newly-written code.

Priorities for path lookups. See also `gtk_binding_set_add_path()`.

Members

GTK_PATH_PRIO_LOWEST	Deprecated
GTK_PATH_PRIO_GTK	Deprecated
GTK_PATH_PRIO_APPLICATION	Deprecated
GTK_PATH_PRIO_THEME	Deprecated
GTK_PATH_PRIO_RC	Deprecated
GTK_PATH_PRIO_HIGHEST	Deprecated

---

enum GtkPathType

`GtkPathType` has been deprecated since version 3.0 and should not be used in newly-written code.

Widget path types. See also `gtk_binding_set_add_path()`.

Members

GTK_PATH_WIDGET	Deprecated
GTK_PATH_WIDGET_CLASS	Deprecated
GTK_PATH_CLASS	Deprecated

GtkStyle

GtkStyle — Deprecated object that holds style information for widgets

Functions

#define	GTK_STYLE_ATTACHED()
GtkStyle *	gtk_style_new ()
GtkStyle *	gtk_style_copy ()
GtkStyle *	gtk_style_attach ()
void	gtk_style_detach ()
gboolean	gtk_style_has_context ()
void	gtk_style_set_background ()
void	gtk_style_apply_default_background ()
gboolean	gtk_style_lookup_color ()
GtkIconSet *	gtk_style_lookup_icon_set ()
GdkPixbuf *	gtk_style_render_icon ()
void	gtk_style_get_style_property ()
void	gtk_style_get_valist ()
void	gtk_style_get ()
void	gtk_paint_arrow ()
void	gtk_paint_box ()
void	gtk_paint_box_gap ()
void	gtk_paint_check ()
void	gtk_paint_diamond ()
void	gtk_paint_extension ()
void	gtk_paint_flat_box ()
void	gtk_paint_focus ()
void	gtk_paint_handle ()
void	gtk_paint_hline ()
void	gtk_paint_option ()
void	gtk_paint_shadow ()
void	gtk_paint_shadow_gap ()
void	gtk_paint_slider ()
void	gtk_paint_spinner ()
void	gtk_paint_tab ()
void	gtk_paint_vline ()
void	gtk_paint_expander ()
void	gtk_paint_layout ()
void	gtk_paint_resize_grip ()
void	gtk_draw_insertion_cursor ()
gboolean	(*GtkRcPropertyParser) ()

Properties

GtkStyleContext *	context	Read / Write / Construct Only

Signals

void	realize	Run First
void	unrealize	Run First

Types and Values

	GtkStyle
struct	GtkStyleClass
enum	GtkExpanderStyle
	GtkRcProperty

Object Hierarchy

```
GObject
└── GtkStyle
```

Includes

```
#include <gtk/gtk.h>
```

Description

A GtkStyle object encapsulates the information that provides the look and feel for a widget.

> In GTK+ 3.0, GtkStyle has been deprecated and replaced by GtkStyleContext.

Each GtkWidget has an associated GtkStyle object that is used when rendering that widget. Also, a GtkStyle holds information for the five possible widget states though not every widget supports all five states; see GtkStateType.

Usually the GtkStyle for a widget is the same as the default style that is set by GTK+ and modified the theme engine.

Usually applications should not need to use or modify the GtkStyle of their widgets.

Functions

GTK_STYLE_ATTACHED()

```
#define GTK_STYLE_ATTACHED(style)       (GTK_STYLE (style)->attach_count > 0)
```

Parameters

style	a GtkStyle.

Returns

whether the style is attached to a window.

gtk_style_new ()

```
GtkStyle *
gtk_style_new (void);
```

`gtk_style_new` has been deprecated since version 3.0 and should not be used in newly-written code.

Use GtkStyleContext

Creates a new GtkStyle.

Returns

a new GtkStyle.

---

gtk_style_copy ()

```
GtkStyle *
gtk_style_copy (GtkStyle *style);
```

`gtk_style_copy` has been deprecated since version 3.0 and should not be used in newly-written code.

Use GtkStyleContext instead

Creates a copy of the passed in GtkStyle object.

Parameters

style	a GtkStyle

Returns

a copy of `style` .

[transfer full]

---

gtk_style_attach ()

```
GtkStyle *
gtk_style_attach (GtkStyle *style,
                  GdkWindow *window);
```

`gtk_style_attach` has been deprecated since version 3.0 and should not be used in newly-written code.

Use `gtk_widget_style_attach()` instead

Attaches a style to a window; this process allocates the colors and creates the GC's for the style - it specializes it to a particular visual. The process may involve the creation of a new style if the style has already been attached to a window with a different style and visual.

Since this function may return a new object, you have to use it in the following way: `style =`

`gtk_style_attach (style, window)`

[skip]

Parameters

style	a GtkStyle.
window	a GdkWindow.

Returns

Either `style` , or a newly-created GtkStyle. If the style is newly created, the style parameter will be unref'ed, and the new style will have a reference count belonging to the caller.

---

gtk_style_detach ()

```
void
gtk_style_detach (GtkStyle *style);
```

`gtk_style_detach` has been deprecated since version 3.0 and should not be used in newly-written code.

Use GtkStyleContext instead

Detaches a style from a window. If the style is not attached to any windows anymore, it is unrealized. See `gtk_style_attach()`.

Parameters

style	a GtkStyle

---

gtk_style_has_context ()

```
gboolean
gtk_style_has_context (GtkStyle *style);
```

`gtk_style_has_context` is deprecated and should not be used in newly-written code.

Returns whether `style` has an associated GtkStyleContext.

Parameters

style	a GtkStyle

Returns

`TRUE` if `style` has a GtkStyleContext

Since: 3.0

---

gtk_style_set_background ()

```
void
gtk_style_set_background (GtkStyle *style,
                          GdkWindow *window,
                          GtkStateType state_type);
```

`gtk_style_set_background` has been deprecated since version 3.0 and should not be used in newly-written code.

Use `gtk_style_context_set_background()` instead

Sets the background of `window` to the background color or pixmap specified by `style` for the given state.

Parameters

style	a GtkStyle
window	a GdkWindow
state_type	a state

---

gtk_style_apply_default_background ()

```
void
gtk_style_apply_default_background (GtkStyle *style,
                                    cairo_t *cr,
                                    GdkWindow *window,
                                    GtkStateType state_type,
                                    gint x,
                                    gint y,
                                    gint width,
                                    gint height);
```

`gtk_style_apply_default_background` has been deprecated since version 3.0 and should not be used in newly-written code.

Use GtkStyleContext instead

---

gtk_style_lookup_color ()

```
gboolean
gtk_style_lookup_color (GtkStyle *style,
                        const gchar *color_name,
                        GdkColor *color);
```

`gtk_style_lookup_color` has been deprecated since version 3.0 and should not be used in newly-written code.

Use `gtk_style_context_lookup_color()` instead

Looks up `color_name` in the style's logical color mappings, filling in `color` and returning `TRUE` if found, otherwise returning `FALSE`. Do not cache the found mapping, because it depends on the GtkStyle and might change when a theme switch occurs.

Parameters

style	a GtkStyle	
color_name	the name of the logical color to look up	
color	the GdkColor to fill in.	[out]

Returns

`TRUE` if the mapping was found.

Since: 2.10

---

gtk_style_lookup_icon_set ()

```
GtkIconSet *
gtk_style_lookup_icon_set (GtkStyle *style,
                           const gchar *stock_id);
```

`gtk_style_lookup_icon_set` has been deprecated since version 3.0 and should not be used in newly-written code.

Use `gtk_style_context_lookup_icon_set()` instead

Looks up `stock_id` in the icon factories associated with `style` and the default icon factory, returning an icon set if found, otherwise `NULL`.

Parameters

style	a GtkStyle
stock_id	an icon name

Returns

icon set of `stock_id` .

[transfer none]

---

gtk_style_render_icon ()

```
GdkPixbuf *
gtk_style_render_icon (GtkStyle *style,
                       const GtkIconSource *source,
                       GtkTextDirection direction,
                       GtkStateType state,
                       GtkIconSize size,
                       GtkWidget *widget,
                       const gchar *detail);
```

`gtk_style_render_icon` has been deprecated since version 3.0 and should not be used in newly-written code.

Use `gtk_render_icon_pixbuf()` instead

Renders the icon specified by `source` at the given `size` according to the given parameters and returns the result in a pixbuf.

Parameters

style	a GtkStyle	
source	the GtkIconSource specifying the icon to render	
direction	a text direction	
state	a state	
size	the size to render the icon at (GtkIconSize). A size of `(GtkIconSize)-1` means render at the size of the source and don't scale.	[type int]
widget	the widget.	[allow-none]
detail	a style detail.	[allow-none]

Returns

a newly-created GdkPixbuf containing the rendered icon.

[transfer full]

---

gtk_style_get_style_property ()

```
void
gtk_style_get_style_property (GtkStyle *style,
                              GType widget_type,
                              const gchar *property_name,
                              GValue *value);
```

`gtk_style_get_style_property` is deprecated and should not be used in newly-written code.

Queries the value of a style property corresponding to a widget class is in the given style.

Parameters

style	a GtkStyle	
widget_type	the GType of a descendant of GtkWidget	
property_name	the name of the style property to get	
value	a GValue where the value of the property being queried will be stored.	[out]

Since: 2.16

---

gtk_style_get_valist ()

```
void
gtk_style_get_valist (GtkStyle *style,
                      GType widget_type,
                      const gchar *first_property_name,
                      va_list var_args);
```

`gtk_style_get_valist` is deprecated and should not be used in newly-written code.

Non-vararg variant of `gtk_style_get()`. Used primarily by language bindings.

Parameters

style	a GtkStyle
widget_type	the GType of a descendant of GtkWidget
first_property_name	the name of the first style property to get
var_args	a va_list of pairs of property names and locations to return the property values, starting with the location for `first_property_name` .

Since: 2.16

---

gtk_style_get ()

```
void
gtk_style_get (GtkStyle *style,
               GType widget_type,
               const gchar *first_property_name,
               ...);
```

`gtk_style_get` is deprecated and should not be used in newly-written code.

Gets the values of a multiple style properties for `widget_type` from `style` .

Parameters

style	a GtkStyle
widget_type	the GType of a descendant of GtkWidget
first_property_name	the name of the first style property to get
...	pairs of property names and locations to return the property values, starting with the location for `first_property_name` , terminated by `NULL`.

Since: 2.16

---

gtk_paint_arrow ()

```
void
gtk_paint_arrow (GtkStyle *style,
                 cairo_t *cr,
                 GtkStateType state_type,
                 GtkShadowType shadow_type,
                 GtkWidget *widget,
                 const gchar *detail,
                 GtkArrowType arrow_type,
                 gboolean fill,
                 gint x,
                 gint y,
                 gint width,
                 gint height);
```

`gtk_paint_arrow` has been deprecated since version 3.0 and should not be used in newly-written code.

Use `gtk_render_arrow()` instead

Draws an arrow in the given rectangle on `cr` using the given parameters. `arrow_type` determines the direction of the arrow.

Parameters

style	a GtkStyle	
cr	a cairo_t	
state_type	a state	
shadow_type	the type of shadow to draw	
widget	the widget.	[allow-none]
detail	a style detail.	[allow-none]
arrow_type	the type of arrow to draw	
fill	`TRUE` if the arrow tip should be filled	
x	x origin of the rectangle to draw the arrow in	
y	y origin of the rectangle to draw the arrow in	
width	width of the rectangle to draw the arrow in	
height	height of the rectangle to draw the arrow in	

---

gtk_paint_box ()

```
void
gtk_paint_box (GtkStyle *style,
               cairo_t *cr,
               GtkStateType state_type,
               GtkShadowType shadow_type,
               GtkWidget *widget,
               const gchar *detail,
               gint x,
               gint y,
               gint width,
```

```
                gint height);
```

`gtk_paint_box` has been deprecated since version 3.0 and should not be used in newly-written code.

Use `gtk_render_frame()` and `gtk_render_background()` instead

Draws a box on `cr` with the given parameters.

Parameters

style	a GtkStyle	
cr	a cairo_t	
state_type	a state	
shadow_type	the type of shadow to draw	
widget	the widget.	[allow-none]
detail	a style detail.	[allow-none]
x	x origin of the box	
y	y origin of the box	
width	the width of the box	
height	the height of the box	

---

gtk_paint_box_gap ()

```
void
gtk_paint_box_gap (GtkStyle *style,
                   cairo_t *cr,
                   GtkStateType state_type,
                   GtkShadowType shadow_type,
                   GtkWidget *widget,
                   const gchar *detail,
                   gint x,
                   gint y,
                   gint width,
                   gint height,
                   GtkPositionType gap_side,
                   gint gap_x,
                   gint gap_width);
```

`gtk_paint_box_gap` has been deprecated since version 3.0 and should not be used in newly-written code.

Use `gtk_render_frame_gap()` instead

Draws a box in `cr` using the given style and state and shadow type, leaving a gap in one side.

Parameters

style	a GtkStyle	
cr	a cairo_t	
state_type	a state	
shadow_type	type of shadow to draw	
widget	the widget.	[allow-none]
detail	a style detail.	[allow-none]
x	x origin of the rectangle	
y	y origin of the rectangle	

width	width of the rectangle
height	width of the rectangle
gap_side	side in which to leave the gap
gap_x	starting position of the gap
gap_width	width of the gap

---

gtk_paint_check ()

```
void
gtk_paint_check (GtkStyle *style,
                 cairo_t *cr,
                 GtkStateType state_type,
                 GtkShadowType shadow_type,
                 GtkWidget *widget,
                 const gchar *detail,
                 gint x,
                 gint y,
                 gint width,
                 gint height);
```

`gtk_paint_check` has been deprecated since version 3.0 and should not be used in newly-written code.

Use `gtk_render_check()` instead

Draws a check button indicator in the given rectangle on `cr` with the given parameters.

Parameters

style	a GtkStyle	
cr	a cairo_t	
state_type	a state	
shadow_type	the type of shadow to draw	
widget	the widget.	[allow-none]
detail	a style detail.	[allow-none]
x	x origin of the rectangle to draw the check in	
y	y origin of the rectangle to draw the check in	
width	the width of the rectangle to draw the check in	
height	the height of the rectangle to draw the check in	

---

gtk_paint_diamond ()

```
void
gtk_paint_diamond (GtkStyle *style,
                   cairo_t *cr,
                   GtkStateType state_type,
                   GtkShadowType shadow_type,
                   GtkWidget *widget,
```

```
                    const gchar *detail,
                    gint x,
                    gint y,
                    gint width,
                    gint height);
```

`gtk_paint_diamond` has been deprecated since version 3.0 and should not be used in newly-written code.

Use cairo instead

Draws a diamond in the given rectangle on `window` using the given parameters.

Parameters

style	a GtkStyle	
cr	a cairo_t	
state_type	a state	
shadow_type	the type of shadow to draw	
widget	the widget.	[allow-none]
detail	a style detail.	[allow-none]
x	x origin of the rectangle to draw the diamond in	
y	y origin of the rectangle to draw the diamond in	
width	width of the rectangle to draw the diamond in	
height	height of the rectangle to draw the diamond in	

---

gtk_paint_extension ()

```
void
gtk_paint_extension (GtkStyle *style,
                     cairo_t *cr,
                     GtkStateType state_type,
                     GtkShadowType shadow_type,
                     GtkWidget *widget,
                     const gchar *detail,
                     gint x,
                     gint y,
                     gint width,
                     gint height,
                     GtkPositionType gap_side);
```

`gtk_paint_extension` has been deprecated since version 3.0 and should not be used in newly-written code.

Use `gtk_render_extension()` instead

Draws an extension, i.e. a notebook tab.

Parameters

style	a GtkStyle
cr	a cairo_t

state_type	a state	
shadow_type	type of shadow to draw	
widget	the widget.	[allow-none]
detail	a style detail.	[allow-none]
x	x origin of the extension	
y	y origin of the extension	
width	width of the extension	
height	width of the extension	
gap_side	the side on to which the extension is attached	

---

gtk_paint_flat_box ()

```
void
gtk_paint_flat_box (GtkStyle *style,
                    cairo_t *cr,
                    GtkStateType state_type,
                    GtkShadowType shadow_type,
                    GtkWidget *widget,
                    const gchar *detail,
                    gint x,
                    gint y,
                    gint width,
                    gint height);
```

`gtk_paint_flat_box` has been deprecated since version 3.0 and should not be used in newly-written code.

Use `gtk_render_frame()` and `gtk_render_background()` instead

Draws a flat box on `cr` with the given parameters.

Parameters

style	a GtkStyle	
cr	a cairo_t	
state_type	a state	
shadow_type	the type of shadow to draw	
widget	the widget.	[allow-none]
detail	a style detail.	[allow-none]
x	x origin of the box	
y	y origin of the box	
width	the width of the box	
height	the height of the box	

---

gtk_paint_focus ()

```
void
gtk_paint_focus (GtkStyle *style,
                 cairo_t *cr,
                 GtkStateType state_type,
                 GtkWidget *widget,
```

```
                const gchar *detail,
                gint x,
                gint y,
                gint width,
                gint height);
```

`gtk_paint_focus` has been deprecated since version 3.0 and should not be used in newly-written code.

Use `gtk_render_focus()` instead

Draws a focus indicator around the given rectangle on `cr` using the given style.

Parameters

style	a GtkStyle	
cr	a cairo_t	
state_type	a state	
widget	the widget.	[allow-none]
detail	a style detail.	[allow-none]
x	the x origin of the rectangle around which to draw a focus indicator	
y	the y origin of the rectangle around which to draw a focus indicator	
width	the width of the rectangle around which to draw a focus indicator	
height	the height of the rectangle around which to draw a focus indicator	

---

gtk_paint_handle ()

```
void
gtk_paint_handle (GtkStyle *style,
                  cairo_t *cr,
                  GtkStateType state_type,
                  GtkShadowType shadow_type,
                  GtkWidget *widget,
                  const gchar *detail,
                  gint x,
                  gint y,
                  gint width,
                  gint height,
                  GtkOrientation orientation);
```

`gtk_paint_handle` has been deprecated since version 3.0 and should not be used in newly-written code.

Use `gtk_render_handle()` instead

Draws a handle as used in GtkHandleBox and GtkPaned.

Parameters

style	a GtkStyle
cr	a cairo_t
state_type	a state

shadow_type	type of shadow to draw	
widget	the widget.	[allow-none]
detail	a style detail.	[allow-none]
x	x origin of the handle	
y	y origin of the handle	
width	with of the handle	
height	height of the handle	
orientation	the orientation of the handle	

---

gtk_paint_hline ()

```
void
gtk_paint_hline (GtkStyle *style,
                 cairo_t *cr,
                 GtkStateType state_type,
                 GtkWidget *widget,
                 const gchar *detail,
                 gint x1,
                 gint x2,
                 gint y);
```

`gtk_paint_hline` has been deprecated since version 3.0 and should not be used in newly-written code.

Use `gtk_render_line()` instead

Draws a horizontal line from (`x1` , `y`) to (`x2` , `y`) in `cr` using the given style and state.

Parameters

style	a GtkStyle	
cr	a caio_t	
state_type	a state	
widget	the widget.	[allow-none]
detail	a style detail.	[allow-none]
x1	the starting x coordinate	
x2	the ending x coordinate	
y	the y coordinate	

---

gtk_paint_option ()

```
void
gtk_paint_option (GtkStyle *style,
                  cairo_t *cr,
                  GtkStateType state_type,
                  GtkShadowType shadow_type,
                  GtkWidget *widget,
                  const gchar *detail,
                  gint x,
                  gint y,
                  gint width,
                  gint height);
```

`gtk_paint_option` has been deprecated since version 3.0 and should not be used in newly-written code.

Use `gtk_render_option()` instead

Draws a radio button indicator in the given rectangle on `cr` with the given parameters.

Parameters

style	a GtkStyle	
cr	a cairo_t	
state_type	a state	
shadow_type	the type of shadow to draw	
widget	the widget.	[allow-none]
detail	a style detail.	[allow-none]
x	x origin of the rectangle to draw the option in	
y	y origin of the rectangle to draw the option in	
width	the width of the rectangle to draw the option in	
height	the height of the rectangle to draw the option in	

---

gtk_paint_shadow ()

```
void
gtk_paint_shadow (GtkStyle *style,
                  cairo_t *cr,
                  GtkStateType state_type,
                  GtkShadowType shadow_type,
                  GtkWidget *widget,
                  const gchar *detail,
                  gint x,
                  gint y,
                  gint width,
                  gint height);
```

`gtk_paint_shadow` has been deprecated since version 3.0 and should not be used in newly-written code.

Use `gtk_render_frame()` instead

Draws a shadow around the given rectangle in `cr` using the given style and state and shadow type.

Parameters

style	a GtkStyle	
cr	a cairo_t	
state_type	a state	
shadow_type	type of shadow to draw	
widget	the widget.	[allow-none]
detail	a style detail.	[allow-none]
x	x origin of the rectangle	
y	y origin of the rectangle	

width	width of the rectangle	
height	width of the rectangle	

gtk_paint_shadow_gap ()

```
void
gtk_paint_shadow_gap (GtkStyle *style,
                      cairo_t *cr,
                      GtkStateType state_type,
                      GtkShadowType shadow_type,
                      GtkWidget *widget,
                      const gchar *detail,
                      gint x,
                      gint y,
                      gint width,
                      gint height,
                      GtkPositionType gap_side,
                      gint gap_x,
                      gint gap_width);
```

`gtk_paint_shadow_gap` has been deprecated since version 3.0 and should not be used in newly-written code.

Use `gtk_render_frame_gap()` instead

Draws a shadow around the given rectangle in `cr` using the given style and state and shadow type, leaving a gap in one side.

Parameters

style	a GtkStyle	
cr	a cairo_t	
state_type	a state	
shadow_type	type of shadow to draw	
widget	the widget.	[allow-none]
detail	a style detail.	[allow-none]
x	x origin of the rectangle	
y	y origin of the rectangle	
width	width of the rectangle	
height	width of the rectangle	
gap_side	side in which to leave the gap	
gap_x	starting position of the gap	
gap_width	width of the gap	

gtk_paint_slider ()

```
void
gtk_paint_slider (GtkStyle *style,
                  cairo_t *cr,
                  GtkStateType state_type,
                  GtkShadowType shadow_type,
                  GtkWidget *widget,
                  const gchar *detail,
```

```
                gint x,
                gint y,
                gint width,
                gint height,
                GtkOrientation orientation);
```

`gtk_paint_slider` has been deprecated since version 3.0 and should not be used in newly-written code.

Use `gtk_render_slider()` instead

Draws a slider in the given rectangle on `cr` using the given style and orientation.

Parameters

style	a GtkStyle	
cr	a cairo_t	
state_type	a state	
shadow_type	a shadow	
widget	the widget.	[allow-none]
detail	a style detail.	[allow-none]
x	the x origin of the rectangle in which to draw a slider	
y	the y origin of the rectangle in which to draw a slider	
width	the width of the rectangle in which to draw a slider	
height	the height of the rectangle in which to draw a slider	
orientation	the orientation to be used	

---

gtk_paint_spinner ()

```
void
gtk_paint_spinner (GtkStyle *style,
                   cairo_t *cr,
                   GtkStateType state_type,
                   GtkWidget *widget,
                   const gchar *detail,
                   guint step,
                   gint x,
                   gint y,
                   gint width,
                   gint height);
```

`gtk_paint_spinner` has been deprecated since version 3.0 and should not be used in newly-written code.

Use `gtk_render_icon()` and the GtkStyleContext you are drawing instead

Draws a spinner on `window` using the given parameters.

Parameters

style	a GtkStyle
cr	a cairo_t

state_type	a state	
widget	the widget (may be `NULL`).	[allow-none]
detail	a style detail (may be `NULL`).	[allow-none]
step	the nth step	
x	the x origin of the rectangle in which to draw the spinner	
y	the y origin of the rectangle in which to draw the spinner	
width	the width of the rectangle in which to draw the spinner	
height	the height of the rectangle in which to draw the spinner	

---

gtk_paint_tab ()

```
void
gtk_paint_tab (GtkStyle *style,
               cairo_t *cr,
               GtkStateType state_type,
               GtkShadowType shadow_type,
               GtkWidget *widget,
               const gchar *detail,
               gint x,
               gint y,
               gint width,
               gint height);
```

`gtk_paint_tab` has been deprecated since version 3.0 and should not be used in newly-written code.

Use cairo instead

Draws an option menu tab (i.e. the up and down pointing arrows) in the given rectangle on `cr` using the given parameters.

Parameters

style	a GtkStyle	
cr	a cairo_t	
state_type	a state	
shadow_type	the type of shadow to draw	
widget	the widget.	[allow-none]
detail	a style detail.	[allow-none]
x	x origin of the rectangle to draw the tab in	
y	y origin of the rectangle to draw the tab in	
width	the width of the rectangle to draw the tab in	
height	the height of the rectangle to draw the tab in	

---

gtk_paint_vline ()

```
void
gtk_paint_vline (GtkStyle *style,
                 cairo_t *cr,
                 GtkStateType state_type,
                 GtkWidget *widget,
                 const gchar *detail,
                 gint y1_,
                 gint y2_,
                 gint x);
```

`gtk_paint_vline` has been deprecated since version 3.0 and should not be used in newly-written code.

Use `gtk_render_line()` instead

Draws a vertical line from (`x` , `y1_`) to (`x` , `y2_`) in `cr` using the given style and state.

Parameters

style	a GtkStyle	
cr	a cairo_t	
state_type	a state	
widget	the widget.	[allow-none]
detail	a style detail.	[allow-none]
y1_	the starting y coordinate	
y2_	the ending y coordinate	
x	the x coordinate	

---

gtk_paint_expander ()

```
void
gtk_paint_expander (GtkStyle *style,
                    cairo_t *cr,
                    GtkStateType state_type,
                    GtkWidget *widget,
                    const gchar *detail,
                    gint x,
                    gint y,
                    GtkExpanderStyle expander_style);
```

`gtk_paint_expander` has been deprecated since version 3.0 and should not be used in newly-written code.

Use `gtk_render_expander()` instead

Draws an expander as used in GtkTreeView. `x` and `y` specify the center the expander. The size of the expander is determined by the “expander-size” style property of `widget` . (If widget is not specified or doesn’t have an “expander-size” property, an unspecified default size will be used, since the caller doesn't have sufficient information to position the expander, this is likely not useful.) The expander is expander_size pixels tall in the collapsed position and expander_size pixels wide in the expanded position.

Parameters

style	a GtkStyle
cr	a cairo_t

state_type	a state	
widget	the widget.	[allow-none]
detail	a style detail.	[allow-none]
x	the x position to draw the expander at	
y	the y position to draw the expander at	
expander_style	the style to draw the expander in; determines whether the expander is collapsed, expanded, or in an intermediate state.	

---

gtk_paint_layout ()

```
void
gtk_paint_layout (GtkStyle *style,
                  cairo_t *cr,
                  GtkStateType state_type,
                  gboolean use_text,
                  GtkWidget *widget,
                  const gchar *detail,
                  gint x,
                  gint y,
                  PangoLayout *layout);
```

`gtk_paint_layout` has been deprecated since version 3.0 and should not be used in newly-written code.

Use `gtk_render_layout()` instead

Draws a layout on `cr` using the given parameters.

Parameters

style	a GtkStyle	
cr	a cairo_t	
state_type	a state	
use_text	whether to use the text or foreground graphics context of `style`	
widget	the widget.	[allow-none]
detail	a style detail.	[allow-none]
x	x origin	
y	y origin	
layout	the layout to draw	

---

gtk_paint_resize_grip ()

```
void
gtk_paint_resize_grip (GtkStyle *style,
                       cairo_t *cr,
                       GtkStateType state_type,
```

```
                        GtkWidget *widget,
                        const gchar *detail,
                        GdkWindowEdge edge,
                        gint x,
                        gint y,
                        gint width,
                        gint height);
```

`gtk_paint_resize_grip` has been deprecated since version 3.0 and should not be used in newly-written code.

Use `gtk_render_handle()` instead

Draws a resize grip in the given rectangle on `cr` using the given parameters.

Parameters

style	a GtkStyle	
cr	a cairo_t	
state_type	a state	
widget	the widget.	[allow-none]
detail	a style detail.	[allow-none]
edge	the edge in which to draw the resize grip	
x	the x origin of the rectangle in which to draw the resize grip	
y	the y origin of the rectangle in which to draw the resize grip	
width	the width of the rectangle in which to draw the resize grip	
height	the height of the rectangle in which to draw the resize grip	

---

gtk_draw_insertion_cursor ()

```
void
gtk_draw_insertion_cursor (GtkWidget *widget,
                           cairo_t *cr,
                           const GdkRectangle *location,
                           gboolean is_primary,
                           GtkTextDirection direction,
                           gboolean draw_arrow);
```

`gtk_draw_insertion_cursor` has been deprecated since version 3.4 and should not be used in newly-written code.

Use `gtk_render_insertion_cursor()` instead.

Draws a text caret on `cr` at `location` . This is not a style function but merely a convenience function for drawing the standard cursor shape.

Parameters

widget	a GtkWidget
cr	cairo context to draw to

location	location where to draw the cursor (`location->width` is ignored)
is_primary	if the cursor should be the primary cursor color.
direction	whether the cursor is left-to-right or right-to-left. Should never be GTK_TEXT_DIR_NONE
draw_arrow	`TRUE` to draw a directional arrow on the cursor. Should be `FALSE` unless the cursor is split.

Since: 3.0

GtkRcPropertyParser ()

```
gboolean
(*GtkRcPropertyParser) (const GParamSpec *pspec,
                        const GString *rc_string,
                        GValue *property_value);
```

Types and Values

GtkStyle

```
typedef struct {
  GdkColor fg[5];
  GdkColor bg[5];
  GdkColor light[5];
  GdkColor dark[5];
  GdkColor mid[5];
  GdkColor text[5];
  GdkColor base[5];
  GdkColor text_aa[5];          /* Halfway between text/base */

  GdkColor black;
  GdkColor white;
  PangoFontDescription *font_desc;

  gint xthickness;
  gint ythickness;

  cairo_pattern_t *background[5];
} GtkStyle;
```

Members

GdkColor `fg`[5];	Set of foreground GdkColor
GdkColor `bg`[5];	Set of background GdkColor
GdkColor `light`[5];	Set of light GdkColor
GdkColor `dark`[5];	Set of dark GdkColor
GdkColor `mid`[5];	Set of mid GdkColor
GdkColor `text`[5];	Set of text GdkColor

GdkColor `base`[5];	Set of base GdkColor
GdkColor `text_aa`[5];	Color halfway between text/base
GdkColor `black`;	GdkColor to use for black
GdkColor `white`;	GdkColor to use for white
PangoFontDescription `*font_desc`;	PangoFontDescription
gint `xthickness`;	Thickness in X direction
gint `ythickness`;	Thickness in Y direction
cairo_pattern_t `*background`[5];	Set of background cairo_pattern_t

---

struct GtkStyleClass

```
struct GtkStyleClass {
  GObjectClass parent_class;


  /* Initialize for a particular visual. style->visual
   * will have been set at this point. Will typically chain
   * to parent.
   */
  void (*realize)              (GtkStyle              *style);

  /* Clean up for a particular visual. Will typically chain
   * to parent.
   */
  void (*unrealize)            (GtkStyle              *style);

  /* Make style an exact duplicate of src.
   */
  void (*copy)                 (GtkStyle              *style,
                                GtkStyle              *src);

  /* Create an empty style of the same type as this style.
   * The default implementation, which does
   * g_object_new (G_OBJECT_TYPE (style), NULL);
   * should work in most cases.
   */
  GtkStyle *(*clone)            (GtkStyle              *style);

  /* Initialize the GtkStyle with the values in the GtkRcStyle.
   * should chain to the parent implementation.
   */
  void     (*init_from_rc)     (GtkStyle              *style,
                                GtkRcStyle            *rc_style);

  void (*set_background)       (GtkStyle              *style,
                                GdkWindow             *window,
                                GtkStateType           state_type);


  GdkPixbuf * (* render_icon)  (GtkStyle              *style,
                                const GtkIconSource   *source,
                                GtkTextDirection       direction,
                                GtkStateType           state,
                                GtkIconSize            size,
                                GtkWidget             *widget,
                                const gchar           *detail);

  /* Drawing functions
```

```
 */
void (*draw_hline)           (GtkStyle              *style,
                              cairo_t               *cr,
                              GtkStateType           state_type,
                              GtkWidget             *widget,
                              const gchar           *detail,
                              gint                   x1,
                              gint                   x2,
                              gint                   y);
void (*draw_vline)           (GtkStyle              *style,
                              cairo_t               *cr,
                              GtkStateType           state_type,
                              GtkWidget             *widget,
                              const gchar           *detail,
                              gint                   y1_,
                              gint                   y2_,
                              gint                   x);
void (*draw_shadow)          (GtkStyle              *style,
                              cairo_t               *cr,
                              GtkStateType           state_type,
                              GtkShadowType          shadow_type,
                              GtkWidget             *widget,
                              const gchar           *detail,
                              gint                   x,
                              gint                   y,
                              gint                   width,
                              gint                   height);
void (*draw_arrow)           (GtkStyle              *style,
                              cairo_t               *cr,
                              GtkStateType           state_type,
                              GtkShadowType          shadow_type,
                              GtkWidget             *widget,
                              const gchar           *detail,
                              GtkArrowType           arrow_type,
                              gboolean               fill,
                              gint                   x,
                              gint                   y,
                              gint                   width,
                              gint                   height);
void (*draw_diamond)         (GtkStyle              *style,
                              cairo_t               *cr,
                              GtkStateType           state_type,
                              GtkShadowType          shadow_type,
                              GtkWidget             *widget,
                              const gchar           *detail,
                              gint                   x,
                              gint                   y,
                              gint                   width,
                              gint                   height);
void (*draw_box)             (GtkStyle              *style,
                              cairo_t               *cr,
                              GtkStateType           state_type,
                              GtkShadowType          shadow_type,
                              GtkWidget             *widget,
                              const gchar           *detail,
                              gint                   x,
                              gint                   y,
                              gint                   width,
                              gint                   height);
void (*draw_flat_box)        (GtkStyle              *style,
                              cairo_t               *cr,
                              GtkStateType           state_type,
```

```
                            GtkShadowType          shadow_type,
                            GtkWidget             *widget,
                            const gchar           *detail,
                            gint                   x,
                            gint                   y,
                            gint                   width,
                            gint                   height);
void (*draw_check)         (GtkStyle              *style,
                            cairo_t               *cr,
                            GtkStateType           state_type,
                            GtkShadowType          shadow_type,
                            GtkWidget             *widget,
                            const gchar           *detail,
                            gint                   x,
                            gint                   y,
                            gint                   width,
                            gint                   height);
void (*draw_option)        (GtkStyle              *style,
                            cairo_t               *cr,
                            GtkStateType           state_type,
                            GtkShadowType          shadow_type,
                            GtkWidget             *widget,
                            const gchar           *detail,
                            gint                   x,
                            gint                   y,
                            gint                   width,
                            gint                   height);
void (*draw_tab)           (GtkStyle              *style,
                            cairo_t               *cr,
                            GtkStateType           state_type,
                            GtkShadowType          shadow_type,
                            GtkWidget             *widget,
                            const gchar           *detail,
                            gint                   x,
                            gint                   y,
                            gint                   width,
                            gint                   height);
void (*draw_shadow_gap)    (GtkStyle              *style,
                            cairo_t               *cr,
                            GtkStateType           state_type,
                            GtkShadowType          shadow_type,
                            GtkWidget             *widget,
                            const gchar           *detail,
                            gint                   x,
                            gint                   y,
                            gint                   width,
                            gint                   height,
                            GtkPositionType        gap_side,
                            gint                   gap_x,
                            gint                   gap_width);
void (*draw_box_gap)       (GtkStyle              *style,
                            cairo_t               *cr,
                            GtkStateType           state_type,
                            GtkShadowType          shadow_type,
                            GtkWidget             *widget,
                            const gchar           *detail,
                            gint                   x,
                            gint                   y,
                            gint                   width,
                            gint                   height,
                            GtkPositionType        gap_side,
                            gint                   gap_x,
                            gint                   gap_width);
```

```
void (*draw_extension)        (GtkStyle              *style,
                               cairo_t               *cr,
                               GtkStateType           state_type,
                               GtkShadowType          shadow_type,
                               GtkWidget             *widget,
                               const gchar           *detail,
                               gint                   x,
                               gint                   y,
                               gint                   width,
                               gint                   height,
                               GtkPositionType        gap_side);
void (*draw_focus)            (GtkStyle              *style,
                               cairo_t               *cr,
                               GtkStateType           state_type,
                               GtkWidget             *widget,
                               const gchar           *detail,
                               gint                   x,
                               gint                   y,
                               gint                   width,
                               gint                   height);
void (*draw_slider)           (GtkStyle              *style,
                               cairo_t               *cr,
                               GtkStateType           state_type,
                               GtkShadowType          shadow_type,
                               GtkWidget             *widget,
                               const gchar           *detail,
                               gint                   x,
                               gint                   y,
                               gint                   width,
                               gint                   height,
                               GtkOrientation         orientation);
void (*draw_handle)           (GtkStyle              *style,
                               cairo_t               *cr,
                               GtkStateType           state_type,
                               GtkShadowType          shadow_type,
                               GtkWidget             *widget,
                               const gchar           *detail,
                               gint                   x,
                               gint                   y,
                               gint                   width,
                               gint                   height,
                               GtkOrientation         orientation);

void (*draw_expander)         (GtkStyle              *style,
                               cairo_t               *cr,
                               GtkStateType           state_type,
                               GtkWidget             *widget,
                               const gchar           *detail,
                               gint                   x,
                               gint                   y,
                               GtkExpanderStyle       expander_style);
void (*draw_layout)           (GtkStyle              *style,
                               cairo_t               *cr,
                               GtkStateType           state_type,
                               gboolean               use_text,
                               GtkWidget             *widget,
                               const gchar           *detail,
                               gint                   x,
                               gint                   y,
                               PangoLayout           *layout);
void (*draw_resize_grip)      (GtkStyle              *style,
                               cairo_t               *cr,
                               GtkStateType           state_type,
```

```
                                GtkWidget              *widget,
                                const gchar            *detail,
                                GdkWindowEdge           edge,
                                gint                    x,
                                gint                    y,
                                gint                    width,
                                gint                    height);
  void (*draw_spinner)         (GtkStyle               *style,
                                cairo_t                *cr,
                                GtkStateType            state_type,
                                GtkWidget              *widget,
                                const gchar            *detail,
                                guint                   step,
                                gint                    x,
                                gint                    y,
                                gint                    width,
                                gint                    height);
};
```

Members

`realize` ()
`unrealize` ()
`copy` ()
`clone` ()
`init_from_rc` ()
`set_background` ()
`render_icon` ()
`draw_hline` ()
`draw_vline` ()
`draw_shadow` ()
`draw_arrow` ()
`draw_diamond` ()
`draw_box` ()
`draw_flat_box` ()
`draw_check` ()
`draw_option` ()
`draw_tab` ()
`draw_shadow_gap` ()
`draw_box_gap` ()
`draw_extension` ()
`draw_focus` ()
`draw_slider` ()
`draw_handle` ()
`draw_expander` ()
`draw_layout` ()
`draw_resize_grip` ()
`draw_spinner` ()

---

enum GtkExpanderStyle

Used to specify the style of the expanders drawn by a GtkTreeView.

Members

GTK_EXPANDER_COLLAPSED	The style used for a collapsed subtree.
GTK_EXPANDER_SEMI_COLLAPSED	Intermediate style used during animation.
GTK_EXPANDER_SEMI_EXPANDED	Intermediate style used during animation.
GTK_EXPANDER_EXPANDED	The style used for an expanded subtree.

GtkRcProperty

```
typedef struct {
  /* quark-ified property identifier like “GtkScrollbar::spacing” */
  GQuark type_name;
  GQuark property_name;

  /* fields similar to GtkSettingsValue */
  gchar *origin;
  GValue value;
} GtkRcProperty;
```

Deprecated

Members

GQuark `type_name`;	quark-ified type identifier
GQuark `property_name`;	quark-ified property identifier like “GtkScrollbar::spacing”
gchar *`origin`;	field similar to one found in GtkSettingsValue
GValue `value`;	field similar to one found in GtkSettingsValue

Property Details

The `“context”` property

```
  “context”                  GtkStyleContext *
```

GtkStyleContext to get style from.

Flags: Read / Write / Construct Only

Signal Details

The "realize" signal

```
void
user_function (GtkStyle *style,
               gpointer  user_data)
```

Emitted when the style has been initialized for a particular visual. Connecting to this signal is probably seldom useful since most of the time applications and widgets only deal with styles that have been already realized.

Parameters

style	the object which received the signal
user_data	user data set when the signal handler was connected.

Flags: Run First

Since: 2.4

The "unrealize" signal

```
void
user_function (GtkStyle *style,
               gpointer  user_data)
```

Emitted when the aspects of the style specific to a particular visual is being cleaned up. A connection to this signal can be useful if a widget wants to cache objects as object data on GtkStyle. This signal provides a convenient place to free such cached objects.

Parameters

style	the object which received the signal
user_data	user data set when the signal handler was connected.

Flags: Run First

Since: 2.4

GtkHScale

GtkHScale — A horizontal slider widget for selecting a value from a range

Functions

GtkWidget *	gtk_hscale_new ()
GtkWidget *	gtk_hscale_new_with_range ()

Types and Values

struct	GtkHScale

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkRange
            └── GtkScale
                └── GtkHScale
```

Implemented Interfaces

GtkHScale implements AtkImplementorIface, GtkBuildable and GtkOrientable.

Includes

```
#include <gtk/gtk.h>
```

Description

The GtkHScale widget is used to allow the user to select a value using a horizontal slider. To create one, use `gtk_hscale_new_with_range()`.

The position to show the current value, and the number of decimal places shown can be set using the parent GtkScale class's functions.

GtkHScale has been deprecated, use GtkScale instead.

Functions

gtk_hscale_new ()

```
GtkWidget *
gtk_hscale_new (GtkAdjustment *adjustment);
```

`gtk_hscale_new` has been deprecated since version 3.2 and should not be used in newly-written code.

Use `gtk_scale_new()` with `GTK_ORIENTATION_HORIZONTAL` instead

Creates a new GtkHScale.

Parameters

adjustment	the GtkAdjustment which sets the range of the scale.	[nullable]

Returns

a new GtkHScale.

---

gtk_hscale_new_with_range ()

```
GtkWidget *
gtk_hscale_new_with_range (gdouble min,
                           gdouble max,
                           gdouble step);
```

`gtk_hscale_new_with_range` has been deprecated since version 3.2 and should not be used in newly-written code.

Use `gtk_scale_new_with_range()` with `GTK_ORIENTATION_HORIZONTAL` instead

Creates a new horizontal scale widget that lets the user input a number between `min` and `max` (including `min` and `max`) with the increment `step` . `step` must be nonzero; it's the distance the slider moves when using the arrow keys to adjust the scale value.

Note that the way in which the precision is derived works best if `step` is a power of ten. If the resulting precision is not suitable for your needs, use `gtk_scale_set_digits()` to correct it.

Parameters

min	minimum value
max	maximum value
step	step increment (tick size) used with keyboard shortcuts

Returns

a new GtkHScale

Types and Values

struct GtkHScale

```
struct GtkHScale;
```

GtkVScale

GtkVScale — A vertical slider widget for selecting a value from a range

Functions

GtkWidget *	gtk_vscale_new ()
GtkWidget *	gtk_vscale_new_with_range ()

Types and Values

struct	GtkVScale

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkRange
            └── GtkScale
                └── GtkVScale
```

Implemented Interfaces

GtkVScale implements AtkImplementorIface, GtkBuildable and GtkOrientable.

Includes

```
#include <gtk/gtk.h>
```

Description

The GtkVScale widget is used to allow the user to select a value using a vertical slider. To create one, use `gtk_hscale_new_with_range()`.

The position to show the current value, and the number of decimal places shown can be set using the parent GtkScale class's functions.

GtkVScale has been deprecated, use GtkScale instead.

Functions

gtk_vscale_new ()

```
GtkWidget *
gtk_vscale_new (GtkAdjustment *adjustment);
```

`gtk_vscale_new` has been deprecated since version 3.2 and should not be used in newly-written code.

Use `gtk_scale_new()` with `GTK_ORIENTATION_VERTICAL` instead

Creates a new GtkVScale.

Parameters

adjustment	the GtkAdjustment which sets the range of the scale.

Returns

a new GtkVScale.

---

gtk_vscale_new_with_range ()

```
GtkWidget *
gtk_vscale_new_with_range (gdouble min,
                           gdouble max,
                           gdouble step);
```

`gtk_vscale_new_with_range` has been deprecated since version 3.2 and should not be used in newly-written code.

Use `gtk_scale_new_with_range()` with `GTK_ORIENTATION_VERTICAL` instead

Creates a new vertical scale widget that lets the user input a number between `min` and `max` (including `min` and `max`) with the increment `step` . `step` must be nonzero; it's the distance the slider moves when using the arrow keys to adjust the scale value.

Note that the way in which the precision is derived works best if `step` is a power of ten. If the resulting precision is not suitable for your needs, use `gtk_scale_set_digits()` to correct it.

Parameters

min	minimum value
max	maximum value
step	step increment (tick size) used with keyboard shortcuts

Returns

a new GtkVScale

Types and Values

struct GtkVScale

```
struct GtkVScale;
```

The GtkVScale struct contains private data only, and should be accessed using the functions below.

GtkTearoffMenuItem

GtkTearoffMenuItem — A menu item used to tear off and reattach its menu

Functions

GtkWidget *	gtk_tearoff_menu_item_new ()

Types and Values

struct	GtkTearoffMenuItem
struct	GtkTearoffMenuItemClass

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkContainer
            └── GtkBin
                └── GtkMenuItem
                    └── GtkTearoffMenuItem
```

Implemented Interfaces

GtkTearoffMenuItem implements AtkImplementorIface, GtkBuildable, GtkActivatable and GtkActionable.

Includes

```
#include <gtk/gtk.h>
```

Description

A GtkTearoffMenuItem is a special GtkMenuItem which is used to tear off and reattach its menu.

When its menu is shown normally, the GtkTearoffMenuItem is drawn as a dotted line indicating that the menu can be torn off. Activating it causes its menu to be torn off and displayed in its own window as a tearoff menu.

When its menu is shown as a tearoff menu, the GtkTearoffMenuItem is drawn as a dotted line which has a left pointing arrow graphic indicating that the tearoff menu can be reattached. Activating it will erase the tearoff menu window.

> GtkTearoffMenuItem is deprecated and should not be used in newly written code. Menus are not meant to be torn around.

Functions

gtk_tearoff_menu_item_new ()

```
GtkWidget *
gtk_tearoff_menu_item_new (void);
```

`gtk_tearoff_menu_item_new` has been deprecated since version 3.4 and should not be used in newly-written code.

GtkTearoffMenuItem is deprecated and should not be used in newly written code.

Creates a new GtkTearoffMenuItem.

Returns

a new GtkTearoffMenuItem.

Types and Values

struct GtkTearoffMenuItem

```
struct GtkTearoffMenuItem;
```

---

struct GtkTearoffMenuItemClass

```
struct GtkTearoffMenuItemClass {
  GtkMenuItemClass parent_class;
};
```

Members

See Also

GtkMenu

GtkColorSelection

GtkColorSelection — Deprecated widget used to select a color

Functions

GtkWidget *	gtk_color_selection_new ()
void	gtk_color_selection_set_has_opacity_control ()
gboolean	gtk_color_selection_get_has_opacity_control ()
void	gtk_color_selection_set_has_palette ()

gboolean	gtk_color_selection_get_has_palette ()
guint16	gtk_color_selection_get_current_alpha ()
void	gtk_color_selection_set_current_alpha ()
void	gtk_color_selection_get_current_color ()
void	gtk_color_selection_set_current_color ()
guint16	gtk_color_selection_get_previous_alpha ()
void	gtk_color_selection_set_previous_alpha ()
void	gtk_color_selection_get_previous_color ()
void	gtk_color_selection_set_previous_color ()
void	gtk_color_selection_get_current_rgba ()
void	gtk_color_selection_set_current_rgba ()
void	gtk_color_selection_get_previous_rgba ()
void	gtk_color_selection_set_previous_rgba ()
gboolean	gtk_color_selection_is_adjusting ()
gboolean	gtk_color_selection_palette_from_string ()
gchar *	gtk_color_selection_palette_to_string ()
void	(*GtkColorSelectionChangePaletteFunc) ()
GtkColorSelectionChangePaletteWithScreenFunc	gtk_color_selection_set_change_palette_with_screen_hook ()
void	(*GtkColorSelectionChangePaletteWithScreenFunc) ()

Properties

guint	current-alpha	Read / Write
GdkColor *	current-color	Read / Write
GdkRGBA *	current-rgba	Read / Write
gboolean	has-opacity-control	Read / Write
gboolean	has-palette	Read / Write

Signals

void	color-changed	Run First

Types and Values

struct	GtkColorSelection
struct	GtkColorSelectionClass

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkContainer
            └── GtkBox
                └── GtkColorSelection
```

Implemented Interfaces

GtkColorSelection implements AtkImplementorIface, GtkBuildable and GtkOrientable.

Includes

```
#include <gtk/gtk.h>
```

Description

The GtkColorSelection is a widget that is used to select a color. It consists of a color wheel and number of sliders and entry boxes for color parameters such as hue, saturation, value, red, green, blue, and opacity. It is found on the standard color selection dialog box GtkColorSelectionDialog.

Functions

gtk_color_selection_new ()

```
GtkWidget *
gtk_color_selection_new (void);
```

`gtk_color_selection_new` is deprecated and should not be used in newly-written code.

Creates a new GtkColorSelection.

Returns

a new GtkColorSelection

---

gtk_color_selection_set_has_opacity_control ()

```
void
gtk_color_selection_set_has_opacity_control
                                (GtkColorSelection *colorsel,
                                 gboolean has_opacity);
```

`gtk_color_selection_set_has_opacity_control` is deprecated and should not be used in newly-written code.

Sets the `colorsel` to use or not use opacity.

Parameters

colorsel	a GtkColorSelection
has_opacity	`TRUE` if `colorsel` can set the opacity, `FALSE` otherwise

---

gtk_color_selection_get_has_opacity_control ()

```
gboolean
gtk_color_selection_get_has_opacity_control
                                (GtkColorSelection *colorsel);
```

`gtk_color_selection_get_has_opacity_control` is deprecated and should not be used in newly-written code.

Determines whether the colorsel has an opacity control.

Parameters

colorsel	a GtkColorSelection

Returns

`TRUE` if the `colorsel` has an opacity control, `FALSE` if it does't

---

gtk_color_selection_set_has_palette ()

```
void
gtk_color_selection_set_has_palette (GtkColorSelection *colorsel,
                                     gboolean has_palette);
```

`gtk_color_selection_set_has_palette` is deprecated and should not be used in newly-written code.

Shows and hides the palette based upon the value of `has_palette` .

Parameters

colorsel	a GtkColorSelection
has_palette	`TRUE` if palette is to be visible, `FALSE` otherwise

---

gtk_color_selection_get_has_palette ()

```
gboolean
gtk_color_selection_get_has_palette (GtkColorSelection *colorsel);
```

`gtk_color_selection_get_has_palette` is deprecated and should not be used in newly-written code.

Determines whether the color selector has a color palette.

Parameters

colorsel	a GtkColorSelection

Returns

`TRUE` if the selector has a palette, `FALSE` if it hasn't

---

gtk_color_selection_get_current_alpha ()

```
guint16
gtk_color_selection_get_current_alpha (GtkColorSelection *colorsel);
```

`gtk_color_selection_get_current_alpha` is deprecated and should not be used in newly-written code.

Returns the current alpha value.

Parameters

colorsel	a GtkColorSelection

Returns

an integer between 0 and 65535

---

gtk_color_selection_set_current_alpha ()

```
void
gtk_color_selection_set_current_alpha (GtkColorSelection *colorsel,
                                       guint16 alpha);
```

`gtk_color_selection_set_current_alpha` is deprecated and should not be used in newly-written code.

Sets the current opacity to be `alpha` .

The first time this is called, it will also set the original opacity to be `alpha` too.

Parameters

colorsel	a GtkColorSelection
alpha	an integer between 0 and 65535

---

gtk_color_selection_get_current_color ()

```
void
gtk_color_selection_get_current_color (GtkColorSelection *colorsel,
                                       GdkColor *color);
```

`gtk_color_selection_get_current_color` has been deprecated since version 3.4 and should not be used in newly-written code.

Use `gtk_color_selection_get_current_rgba()` instead.

Sets `color` to be the current color in the GtkColorSelection widget.

Parameters

colorsel	a GtkColorSelection	
color	a GdkColor to fill in with the current color.	[out]

---

gtk_color_selection_set_current_color ()

```
void
gtk_color_selection_set_current_color (GtkColorSelection *colorsel,
                                       const GdkColor *color);
```

`gtk_color_selection_set_current_color` has been deprecated since version 3.4 and should not be used in newly-written code.

Use `gtk_color_selection_set_current_rgba()` instead.

Sets the current color to be `color` .

The first time this is called, it will also set the original color to be `color` too.

Parameters

colorsel	a GtkColorSelection	
color	a GdkColor to set the current color with	

---

gtk_color_selection_get_previous_alpha ()

```
guint16
gtk_color_selection_get_previous_alpha
                                (GtkColorSelection *colorsel);
```

`gtk_color_selection_get_previous_alpha` is deprecated and should not be used in newly-written code.

Returns the previous alpha value.

Parameters

colorsel	a GtkColorSelection	

Returns

an integer between 0 and 65535

---

gtk_color_selection_set_previous_alpha ()

```
void
gtk_color_selection_set_previous_alpha
                                (GtkColorSelection *colorsel,
                                 guint16 alpha);
```

`gtk_color_selection_set_previous_alpha` is deprecated and should not be used in newly-written code.

Sets the “previous” alpha to be `alpha` .

This function should be called with some hesitations, as it might seem confusing to have that alpha change.

Parameters

colorsel	a GtkColorSelection	
alpha	an integer between 0 and 65535	

---

gtk_color_selection_get_previous_color ()

```
void
gtk_color_selection_get_previous_color
                                (GtkColorSelection *colorsel,
                                 GdkColor *color);
```

`gtk_color_selection_get_previous_color` has been deprecated since version 3.4 and should not be used in newly-written code.

Use `gtk_color_selection_get_previous_rgba()` instead.

Fills `color` in with the original color value.

Parameters

colorsel	a GtkColorSelection	
color	a GdkColor to fill in with the original color value.	[out]

---

gtk_color_selection_set_previous_color ()

```
void
gtk_color_selection_set_previous_color
                                (GtkColorSelection *colorsel,
                                 const GdkColor *color);
```

`gtk_color_selection_set_previous_color` has been deprecated since version 3.4 and should not be used in newly-written code.

Use `gtk_color_selection_set_previous_rgba()` instead.

Sets the “previous” color to be `color` .

This function should be called with some hesitations, as it might seem confusing to have that color change. Calling `gtk_color_selection_set_current_color()` will also set this color the first time it is called.

Parameters

colorsel	a GtkColorSelection
color	a GdkColor to set the previous color with

---

gtk_color_selection_get_current_rgba ()

```
void
gtk_color_selection_get_current_rgba (GtkColorSelection *colorsel,
                                      GdkRGBA *rgba);
```

`gtk_color_selection_get_current_rgba` is deprecated and should not be used in newly-written code.

Sets `rgba` to be the current color in the GtkColorSelection widget.

Parameters

colorsel	a GtkColorSelection	
rgba	a GdkRGBA to fill in with the current color.	[out]

Since: 3.0

---

gtk_color_selection_set_current_rgba ()

```
void
gtk_color_selection_set_current_rgba (GtkColorSelection *colorsel,
                                      const GdkRGBA *rgba);
```

`gtk_color_selection_set_current_rgba` is deprecated and should not be used in newly-written code.

Sets the current color to be `rgba` .

The first time this is called, it will also set the original color to be `rgba` too.

Parameters

colorsel	a GtkColorSelection
rgba	A GdkRGBA to set the current color with

Since: 3.0

---

gtk_color_selection_get_previous_rgba ()

```
void
gtk_color_selection_get_previous_rgba (GtkColorSelection *colorsel,
                                       GdkRGBA *rgba);
```

`gtk_color_selection_get_previous_rgba` is deprecated and should not be used in newly-written code.

Fills `rgba` in with the original color value.

Parameters

colorsel	a GtkColorSelection	
rgba	a GdkRGBA to fill in with the original color value.	[out]

Since: 3.0

---

gtk_color_selection_set_previous_rgba ()

```
void
gtk_color_selection_set_previous_rgba (GtkColorSelection *colorsel,
                                       const GdkRGBA *rgba);
```

`gtk_color_selection_set_previous_rgba` is deprecated and should not be used in newly-written code.

Sets the “previous” color to be `rgba` .

This function should be called with some hesitations, as it might seem confusing to have that color change. Calling `gtk_color_selection_set_current_rgba()` will also set this color the first time it is called.

Parameters

colorsel	a GtkColorSelection	
rgba	a GdkRGBA to set the previous color with	

Since: 3.0

---

gtk_color_selection_is_adjusting ()

```
gboolean
gtk_color_selection_is_adjusting (GtkColorSelection *colorsel);
```

`gtk_color_selection_is_adjusting` is deprecated and should not be used in newly-written code.

Gets the current state of the `colorsel` .

Parameters

colorsel	a GtkColorSelection	

Returns

`TRUE` if the user is currently dragging a color around, and `FALSE` if the selection has stopped

---

gtk_color_selection_palette_from_string ()

```
gboolean
gtk_color_selection_palette_from_string
                                (const gchar *str,
                                 GdkColor **colors,
                                 gint *n_colors);
```

`gtk_color_selection_palette_from_string` is deprecated and should not be used in newly-written code.

Parses a color palette string; the string is a colon-separated list of color names readable by `gdk_color_parse()`.

Parameters

str	a string encoding a color palette	
colors	return location for allocated array of GdkColor.	[out][array length=n_colors]
n_colors	return location for length of array	

Returns

`TRUE` if a palette was successfully parsed

---

gtk_color_selection_palette_to_string ()

```
gchar *
gtk_color_selection_palette_to_string (const GdkColor *colors,
                                       gint n_colors);
```

`gtk_color_selection_palette_to_string` is deprecated and should not be used in newly-written code.

Encodes a palette as a string, useful for persistent storage.

Parameters

colors	an array of colors.	[array length=n_colors]
n_colors	length of the array	

Returns

allocated string encoding the palette

GtkColorSelectionChangePaletteFunc ()

```
void
(*GtkColorSelectionChangePaletteFunc) (const GdkColor *colors,
                                       gint n_colors);
```

`GtkColorSelectionChangePaletteFunc` has been deprecated since version 3.4 and should not be used in newly-written code.

Parameters

colors	Array of colors.	[array length=n_colors]
n_colors	Number of colors in the array	

---

gtk_color_selection_set_change_palette_with_screen_hook ()

```
GtkColorSelectionChangePaletteWithScreenFunc
gtk_color_selection_set_change_palette_with_screen_hook
                                (GtkColorSelectionChangePaletteWithScreenFunc func);
```

`gtk_color_selection_set_change_palette_with_screen_hook` is deprecated and should not be used in newly-written code.

Installs a global function to be called whenever the user tries to modify the palette in a color selection.

This function should save the new palette contents, and update the "gtk-color-palette" GtkSettings property so all GtkColorSelection widgets will be modified.

[skip]

Parameters

func	a function to call when the custom palette needs saving

Returns

the previous change palette hook (that was replaced)

Since: 2.2

GtkColorSelectionChangePaletteWithScreenFunc ()

```
void
(*GtkColorSelectionChangePaletteWithScreenFunc)
                                (GdkScreen *screen,
                                 const GdkColor *colors,
                                 gint n_colors);
```

`GtkColorSelectionChangePaletteWithScreenFunc` has been deprecated since version 3.4 and should not be used in newly-written code.

Parameters

colors	Array of colors.	[array length=n_colors]

n_colors	Number of colors in the array

Since: 2.2

Types and Values

struct GtkColorSelection

```
struct GtkColorSelection;
```

---

struct GtkColorSelectionClass

```
struct GtkColorSelectionClass {
  GtkBoxClass parent_class;

  void (*color_changed) (GtkColorSelection *color_selection);
};
```

Members

`color_changed` ()

Property Details

The “current-alpha” property

“current-alpha”	guint

The current opacity value (0 fully transparent, 65535 fully opaque).

Flags: Read / Write

Allowed values: <= 65535

Default value: 65535

The “current-color” property

“current-color”	GdkColor *

The current GdkColor color.

`GtkColorSelection:current-color` has been deprecated since version 3.4 and should not be used in newly-written code.

Use “current-rgba” instead.

Flags: Read / Write

The “current-rgba” property

```
  “current-rgba”             GdkRGBA *
```

The current RGBA color.

Flags: Read / Write

Since: 3.0

The “has-opacity-control” property

```
  “has-opacity-control”      gboolean
```

Whether the color selector should allow setting opacity.

Flags: Read / Write

Default value: FALSE

The “has-palette” property

```
  “has-palette”              gboolean
```

Whether a palette should be used.

Flags: Read / Write

Default value: FALSE

Signal Details

The “color-changed” signal

```
void
user_function (GtkColorSelection *colorselection,
               gpointer           user_data)
```

This signal is emitted when the color changes in the GtkColorSelection according to its update policy.

Parameters

colorselection	the object which received the signal.
user_data	user data set when the signal handler was connected.

Flags: Run First

GtkColorSelectionDialog

GtkColorSelectionDialog — Deprecated dialog box for selecting a color

Functions

GtkWidget *	gtk_color_selection_dialog_new ()
GtkWidget *	gtk_color_selection_dialog_get_color_selection ()

Properties

GtkWidget *	cancel-button	Read
GtkWidget *	color-selection	Read
GtkWidget *	help-button	Read
GtkWidget *	ok-button	Read

Types and Values

struct	GtkColorSelectionDialog

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkContainer
            └── GtkBin
                └── GtkWindow
                    └── GtkDialog
                        └── GtkColorSelectionDialog
```

Implemented Interfaces

GtkColorSelectionDialog implements AtkImplementorIface and GtkBuildable.

Includes

```
#include <gtk/gtk.h>
```

Description

The GtkColorSelectionDialog provides a standard dialog which allows the user to select a color much like the GtkFileChooserDialog provides a standard dialog for file selection.

Use `gtk_color_selection_dialog_get_color_selection()` to get the GtkColorSelection widget contained within the dialog. Use this widget and its `gtk_color_selection_get_current_color()` function to gain access to the selected color. Connect a handler for this widget's "color-changed" signal to be notified when the color changes.

GtkColorSelectionDialog as GtkBuildable

The GtkColorSelectionDialog implementation of the GtkBuildable interface exposes the embedded GtkColorSelection as internal child with the name “color_selection”. It also exposes the buttons with the names “ok_button”, “cancel_button” and “help_button”.

Functions

gtk_color_selection_dialog_new ()

```
GtkWidget *
gtk_color_selection_dialog_new (const gchar *title);
```

`gtk_color_selection_dialog_new` is deprecated and should not be used in newly-written code.

Creates a new GtkColorSelectionDialog.

Parameters

title	a string containing the title text for the dialog.

Returns

a GtkColorSelectionDialog.

---

gtk_color_selection_dialog_get_color_selection ()

```
GtkWidget *
gtk_color_selection_dialog_get_color_selection
                                (GtkColorSelectionDialog *colorsel);
```

`gtk_color_selection_dialog_get_color_selection` is deprecated and should not be used in newly-written code.

Retrieves the GtkColorSelection widget embedded in the dialog.

Parameters

colorsel	a GtkColorSelectionDialog

Returns

the embedded GtkColorSelection.

[transfer none]

Since: 2.14

Types and Values

struct GtkColorSelectionDialog

```
struct GtkColorSelectionDialog;
```

Property Details

The “cancel-button” property

```
  “cancel-button”            GtkWidget *
```

The cancel button of the dialog.

Flags: Read

The “color-selection” property

```
  “color-selection”          GtkWidget *
```

The color selection embedded in the dialog.

Flags: Read

The “help-button” property

```
  “help-button”              GtkWidget *
```

The help button of the dialog.

Flags: Read

The “ok-button” property

```
  “ok-button”                GtkWidget *
```

The OK button of the dialog.

Flags: Read

GtkHSV

GtkHSV — A “color wheel” widget

Functions

GtkWidget *	gtk_hsv_new ()
void	gtk_hsv_set_color ()
void	gtk_hsv_get_color ()
void	gtk_hsv_set_metrics ()
void	gtk_hsv_get_metrics ()
gboolean	gtk_hsv_is_adjusting ()
void	gtk_hsv_to_rgb ()
void	gtk_rgb_to_hsv ()

Signals

void	changed	Run First
void	move	Action

Types and Values

struct	GtkHSV

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkHSV
```

Implemented Interfaces

GtkHSV implements AtkImplementorIface and GtkBuildable.

Includes

```
#include <gtk/gtk.h>
```

Description

GtkHSV is the “color wheel” part of a complete color selector widget. It allows to select a color by determining its HSV components in an intuitive way. Moving the selection around the outer ring changes the hue, and moving the selection point inside the inner triangle changes value and saturation.

GtkHSV has been deprecated together with GtkColorSelection, where it was used.

Functions

gtk_hsv_new ()

```
GtkWidget *
gtk_hsv_new (void);
```

`gtk_hsv_new` is deprecated and should not be used in newly-written code.

Creates a new HSV color selector.

Returns

A newly-created HSV color selector.

Since: 2.14

---

gtk_hsv_set_color ()

```
void
gtk_hsv_set_color (GtkHSV *hsv,
                   double h,
                   double s,
                   double v);
```

`gtk_hsv_set_color` is deprecated and should not be used in newly-written code.

Sets the current color in an HSV color selector. Color component values must be in the [0.0, 1.0] range.

Parameters

hsv	An HSV color selector	
h	Hue	
s	Saturation	
v	Value	

Since: 2.14

---

gtk_hsv_get_color ()

```
void
gtk_hsv_get_color (GtkHSV *hsv,
                   gdouble *h,
                   gdouble *s,
                   gdouble *v);
```

`gtk_hsv_get_color` is deprecated and should not be used in newly-written code.

Queries the current color in an HSV color selector. Returned values will be in the [0.0, 1.0] range.

Parameters

hsv	An HSV color selector	
h	Return value for the hue.	[out]
s	Return value for the saturation.	[out]

v	Return value for the value.	[out]

Since: 2.14

---

gtk_hsv_set_metrics ()

```
void
gtk_hsv_set_metrics (GtkHSV *hsv,
                     gint size,
                     gint ring_width);
```

`gtk_hsv_set_metrics` is deprecated and should not be used in newly-written code.

Sets the size and ring width of an HSV color selector.

Parameters

hsv	An HSV color selector
size	Diameter for the hue ring
ring_width	Width of the hue ring

Since: 2.14

---

gtk_hsv_get_metrics ()

```
void
gtk_hsv_get_metrics (GtkHSV *hsv,
                     gint *size,
                     gint *ring_width);
```

`gtk_hsv_get_metrics` is deprecated and should not be used in newly-written code.

Queries the size and ring width of an HSV color selector.

Parameters

hsv	An HSV color selector	
size	Return value for the diameter of the hue ring.	[out]
ring_width	Return value for the width of the hue ring.	[out]

Since: 2.14

---

gtk_hsv_is_adjusting ()

```
gboolean
gtk_hsv_is_adjusting (GtkHSV *hsv);
```

`gtk_hsv_is_adjusting` is deprecated and should not be used in newly-written code.

An HSV color selector can be said to be adjusting if multiple rapid changes are being made to its value, for example, when the user is adjusting the value with the mouse. This function queries whether the HSV color

selector is being adjusted or not.

Parameters

hsv	A GtkHSV

Returns

`TRUE` if clients can ignore changes to the color value, since they may be transitory, or `FALSE` if they should consider the color value status to be final.

Since: 2.14

---

gtk_hsv_to_rgb ()

```
void
gtk_hsv_to_rgb (gdouble h,
                gdouble s,
                gdouble v,
                gdouble *r,
                gdouble *g,
                gdouble *b);
```

Converts a color from HSV space to RGB.

Input values must be in the [0.0, 1.0] range; output values will be in the same range.

Parameters

h	Hue	
s	Saturation	
v	Value	
r	Return value for the red component.	[out]
g	Return value for the green component.	[out]
b	Return value for the blue component.	[out]

Since: 2.14

---

gtk_rgb_to_hsv ()

```
void
gtk_rgb_to_hsv (gdouble r,
                gdouble g,
                gdouble b,
                gdouble *h,
                gdouble *s,
                gdouble *v);
```

Converts a color from RGB space to HSV.

Input values must be in the [0.0, 1.0] range; output values will be in the same range.

Parameters

r	Red	
g	Green	
b	Blue	
h	Return value for the hue component.	[out]
s	Return value for the saturation component.	[out]
v	Return value for the value component.	[out]

Since: 2.14

Types and Values

struct GtkHSV

```
struct GtkHSV;
```

Signal Details

The "changed" signal

```
void
user_function (GtkHSV  *hsv,
               gpointer user_data)
```

Flags: Run First

The "move" signal

```
void
user_function (GtkHSV          *hsv,
               GtkDirectionType arg1,
               gpointer         user_data)
```

Flags: Action

See Also

GtkColorSelection, GtkColorSelectionDialog

GtkFontSelection

GtkFontSelection — Deprecated widget for selecting fonts

Functions

GtkWidget *	gtk_font_selection_new ()
gchar *	gtk_font_selection_get_font_name ()
gboolean	gtk_font_selection_set_font_name ()
const gchar *	gtk_font_selection_get_preview_text ()
void	gtk_font_selection_set_preview_text ()
PangoFontFace *	gtk_font_selection_get_face ()
GtkWidget *	gtk_font_selection_get_face_list ()
PangoFontFamily *	gtk_font_selection_get_family ()
gint	gtk_font_selection_get_size ()
GtkWidget *	gtk_font_selection_get_family_list ()
GtkWidget *	gtk_font_selection_get_preview_entry ()
GtkWidget *	gtk_font_selection_get_size_entry ()
GtkWidget *	gtk_font_selection_get_size_list ()

Properties

gchar *	font-name	Read / Write
gchar *	preview-text	Read / Write

Types and Values

struct	GtkFontSelection

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkContainer
            └── GtkBox
                └── GtkFontSelection
```

Implemented Interfaces

GtkFontSelection implements AtkImplementorIface, GtkBuildable and GtkOrientable.

Includes

```
#include <gtk/gtk.h>
```

Description

The GtkFontSelection widget lists the available fonts, styles and sizes, allowing the user to select a font. It is used in the GtkFontSelectionDialog widget to provide a dialog box for selecting fonts.

To set the font which is initially selected, use `gtk_font_selection_set_font_name()`.

To get the selected font use `gtk_font_selection_get_font_name()`.

To change the text which is shown in the preview area, use `gtk_font_selection_set_preview_text()`.

In GTK+ 3.2, GtkFontSelection has been deprecated in favor of GtkFontChooser.

Functions

gtk_font_selection_new ()

```
GtkWidget *
gtk_font_selection_new (void);
```

`gtk_font_selection_new` has been deprecated since version 3.2 and should not be used in newly-written code.

Use GtkFontChooserWidget instead

Creates a new GtkFontSelection.

Returns

a new GtkFontSelection

---

gtk_font_selection_get_font_name ()

```
gchar *
gtk_font_selection_get_font_name (GtkFontSelection *fontsel);
```

`gtk_font_selection_get_font_name` has been deprecated since version 3.2 and should not be used in newly-written code.

Use GtkFontChooser

Gets the currently-selected font name.

Note that this can be a different string than what you set with `gtk_font_selection_set_font_name()`, as the font selection widget may normalize font names and thus return a string with a different structure. For example, “Helvetica Italic Bold 12” could be normalized to “Helvetica Bold Italic 12”. Use `pango_font_description_equal()` if you want to compare two font descriptions.

Parameters

fontsel	a GtkFontSelection

Returns

A string with the name of the current font, or `NULL` if no font is selected. You must free this string with `g_free()`.

---

gtk_font_selection_set_font_name ()

```
gboolean
gtk_font_selection_set_font_name (GtkFontSelection *fontsel,
                                  const gchar *fontname);
```

`gtk_font_selection_set_font_name` has been deprecated since version 3.2 and should not be used in newly-written code.

Use GtkFontChooser

Sets the currently-selected font.

Note that the `fontsel` needs to know the screen in which it will appear for this to work; this can be guaranteed by simply making sure that the `fontsel` is inserted in a toplevel window before you call this function.

Parameters

fontsel	a GtkFontSelection
fontname	a font name like “Helvetica 12” or “Times Bold 18”

Returns

`TRUE` if the font could be set successfully; `FALSE` if no such font exists or if the `fontsel` doesn’t belong to a particular screen yet.

---

gtk_font_selection_get_preview_text ()

```
const gchar *
gtk_font_selection_get_preview_text (GtkFontSelection *fontsel);
```

`gtk_font_selection_get_preview_text` has been deprecated since version 3.2 and should not be used in newly-written code.

Use GtkFontChooser

Gets the text displayed in the preview area.

Parameters

fontsel	a GtkFontSelection

Returns

the text displayed in the preview area. This string is owned by the widget and should not be modified or freed

---

gtk_font_selection_set_preview_text ()

```
void
gtk_font_selection_set_preview_text (GtkFontSelection *fontsel,
                                     const gchar *text);
```

`gtk_font_selection_set_preview_text` has been deprecated since version 3.2 and should not be used in newly-written code.

Use GtkFontChooser

Sets the text displayed in the preview area. The `text` is used to show how the selected font looks.

Parameters

fontsel	a GtkFontSelection
text	the text to display in the preview area

---

gtk_font_selection_get_face ()

```
PangoFontFace *
gtk_font_selection_get_face (GtkFontSelection *fontsel);
```

`gtk_font_selection_get_face` has been deprecated since version 3.2 and should not be used in newly-written code.

Use GtkFontChooser

Gets the PangoFontFace representing the selected font group details (i.e. family, slant, weight, width, etc).

Parameters

fontsel	a GtkFontSelection

Returns

A PangoFontFace representing the selected font group details. The returned object is owned by `fontsel` and must not be modified or freed.

[transfer none]

Since: 2.14

---

gtk_font_selection_get_face_list ()

```
GtkWidget *
gtk_font_selection_get_face_list (GtkFontSelection *fontsel);
```

`gtk_font_selection_get_face_list` has been deprecated since version 3.2 and should not be used in newly-written code.

Use GtkFontChooser

This returns the GtkTreeView which lists all styles available for the selected font. For example, “Regular”, “Bold”, etc.

Parameters

fontsel	a GtkFontSelection

Returns

A GtkWidget that is part of `fontsel` .

[transfer none]

Since: 2.14

---

gtk_font_selection_get_family ()

```
PangoFontFamily *
gtk_font_selection_get_family (GtkFontSelection *fontsel);
```

`gtk_font_selection_get_family` has been deprecated since version 3.2 and should not be used in newly-written code.

Use GtkFontChooser

Gets the PangoFontFamily representing the selected font family.

Parameters

fontsel	a GtkFontSelection

Returns

A PangoFontFamily representing the selected font family. Font families are a collection of font faces. The returned object is owned by `fontsel` and must not be modified or freed.

[transfer none]

Since: 2.14

---

gtk_font_selection_get_size ()

```
gint
gtk_font_selection_get_size (GtkFontSelection *fontsel);
```

`gtk_font_selection_get_size` has been deprecated since version 3.2 and should not be used in newly-written code.

Use GtkFontChooser

The selected font size.

Parameters

fontsel	a GtkFontSelection

Returns

A n integer representing the selected font size, or -1 if no font size is selected.

Since: 2.14

---

gtk_font_selection_get_family_list ()

```
GtkWidget *
gtk_font_selection_get_family_list (GtkFontSelection *fontsel);
```

`gtk_font_selection_get_family_list` has been deprecated since version 3.2 and should not be used in newly-written code.

Use GtkFontChooser

This returns the GtkTreeView that lists font families, for example, “Sans”, “Serif”, etc.

Parameters

fontsel	a GtkFontSelection

Returns

A GtkWidget that is part of `fontsel` .

[transfer none]

Since: 2.14

---

gtk_font_selection_get_preview_entry ()

```
GtkWidget *
gtk_font_selection_get_preview_entry (GtkFontSelection *fontsel);
```

`gtk_font_selection_get_preview_entry` has been deprecated since version 3.2 and should not be used in

newly-written code.

Use GtkFontChooser

This returns the GtkEntry used to display the font as a preview.

Parameters

fontsel	a GtkFontSelection

Returns

A GtkWidget that is part of `fontsel` .

[transfer none]

Since: 2.14

---

gtk_font_selection_get_size_entry ()

```
GtkWidget *
gtk_font_selection_get_size_entry (GtkFontSelection *fontsel);
```

`gtk_font_selection_get_size_entry` has been deprecated since version 3.2 and should not be used in newly-written code.

Use GtkFontChooser

This returns the GtkEntry used to allow the user to edit the font number manually instead of selecting it from the list of font sizes.

Parameters

fontsel	a GtkFontSelection

Returns

A GtkWidget that is part of `fontsel` .

[transfer none]

Since: 2.14

---

gtk_font_selection_get_size_list ()

```
GtkWidget *
gtk_font_selection_get_size_list (GtkFontSelection *fontsel);
```

`gtk_font_selection_get_size_list` has been deprecated since version 3.2 and should not be used in newly-written code.

Use GtkFontChooser

This returns the GtkTreeView used to list font sizes.

Parameters

fontsel	a GtkFontSelection

Returns

A GtkWidget that is part of `fontsel`.

[transfer none]

Since: 2.14

Types and Values

struct GtkFontSelection

```
struct GtkFontSelection;
```

Property Details

The `"font-name"` property

```
  "font-name"                gchar *
```

The string that represents this font.

Flags: Read / Write

Default value: "Sans 10"

The `"preview-text"` property

```
  "preview-text"             gchar *
```

The text to display in order to demonstrate the selected font.

Flags: Read / Write

Default value: "abcdefghijk ABCDEFGHIJK"

See Also

GtkFontSelectionDialog, GtkFontChooser

GtkFontSelectionDialog

GtkFontSelectionDialog — Deprecated dialog box for selecting fonts

Functions

GtkWidget *	gtk_font_selection_dialog_new ()
gchar *	gtk_font_selection_dialog_get_font_name ()
gboolean	gtk_font_selection_dialog_set_font_name ()
const gchar *	gtk_font_selection_dialog_get_preview_text ()
void	gtk_font_selection_dialog_set_preview_text ()
GtkWidget *	gtk_font_selection_dialog_get_cancel_button ()
GtkWidget *	gtk_font_selection_dialog_get_ok_button ()
GtkWidget *	gtk_font_selection_dialog_get_font_selection ()

Types and Values

struct	GtkFontSelectionDialog

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkContainer
            └── GtkBin
                └── GtkWindow
                    └── GtkDialog
                        └── GtkFontSelectionDialog
```

Implemented Interfaces

GtkFontSelectionDialog implements AtkImplementorIface and GtkBuildable.

Includes

```
#include <gtk/gtk.h>
```

Description

The GtkFontSelectionDialog widget is a dialog box for selecting a font.

To set the font which is initially selected, use `gtk_font_selection_dialog_set_font_name()`.

To get the selected font use `gtk_font_selection_dialog_get_font_name()`.

To change the text which is shown in the preview area, use `gtk_font_selection_dialog_set_preview_text()`.

In GTK+ 3.2, GtkFontSelectionDialog has been deprecated in favor of GtkFontChooserDialog.

GtkFontSelectionDialog as GtkBuildable

The GtkFontSelectionDialog implementation of the GtkBuildable interface exposes the embedded GtkFontSelection as internal child with the name “font_selection”. It also exposes the buttons with the names “ok_button”, “cancel_button” and “apply_button”.

Functions

gtk_font_selection_dialog_new ()

```
GtkWidget *
gtk_font_selection_dialog_new (const gchar *title);
```

`gtk_font_selection_dialog_new` has been deprecated since version 3.2 and should not be used in newly-written code.

Use GtkFontChooserDialog

Creates a new GtkFontSelectionDialog.

Parameters

title	the title of the dialog window

Returns

a new GtkFontSelectionDialog

---

gtk_font_selection_dialog_get_font_name ()

```
gchar *
gtk_font_selection_dialog_get_font_name
                                (GtkFontSelectionDialog *fsd);
```

`gtk_font_selection_dialog_get_font_name` has been deprecated since version 3.2 and should not be used in newly-written code.

Use GtkFontChooserDialog

Gets the currently-selected font name.

Note that this can be a different string than what you set with `gtk_font_selection_dialog_set_font_name()`, as the font selection widget may normalize font names and thus return a string with a different structure. For example, “Helvetica Italic Bold 12” could be normalized to “Helvetica Bold Italic 12”. Use `pango_font_description_equal()` if you want to compare two font descriptions.

Parameters

fsd	a GtkFontSelectionDialog

Returns

A string with the name of the current font, or `NULL` if no font is selected. You must free this string with `g_free()`.

---

gtk_font_selection_dialog_set_font_name ()

```
gboolean
gtk_font_selection_dialog_set_font_name
                                (GtkFontSelectionDialog *fsd,
                                 const gchar *fontname);
```

`gtk_font_selection_dialog_set_font_name` has been deprecated since version 3.2 and should not be used in newly-written code.

Use GtkFontChooserDialog

Sets the currently selected font.

Parameters

fsd	a GtkFontSelectionDialog
fontname	a font name like “Helvetica 12” or “Times Bold 18”

Returns

`TRUE` if the font selected in `fsd` is now the `fontname` specified, `FALSE` otherwise.

---

gtk_font_selection_dialog_get_preview_text ()

```
const gchar *
gtk_font_selection_dialog_get_preview_text
                                (GtkFontSelectionDialog *fsd);
```

`gtk_font_selection_dialog_get_preview_text` has been deprecated since version 3.2 and should not be used in newly-written code.

Use GtkFontChooserDialog

Gets the text displayed in the preview area.

Parameters

fsd	a GtkFontSelectionDialog

Returns

the text displayed in the preview area. This string is owned by the widget and should not be modified or freed

---

gtk_font_selection_dialog_set_preview_text ()

```
void
gtk_font_selection_dialog_set_preview_text
                                (GtkFontSelectionDialog *fsd,
                                 const gchar *text);
```

`gtk_font_selection_dialog_set_preview_text` has been deprecated since version 3.2 and should not be used in newly-written code.

Use GtkFontChooserDialog

Sets the text displayed in the preview area.

Parameters

fsd	a GtkFontSelectionDialog
text	the text to display in the preview area

---

gtk_font_selection_dialog_get_cancel_button ()

```
GtkWidget *
gtk_font_selection_dialog_get_cancel_button
                                (GtkFontSelectionDialog *fsd);
```

`gtk_font_selection_dialog_get_cancel_button` has been deprecated since version 3.2 and should not be used in newly-written code.

Use GtkFontChooserDialog

Gets the “Cancel” button.

Parameters

fsd	a GtkFontSelectionDialog

Returns

the GtkWidget used in the dialog for the “Cancel” button.

[transfer none]

Since: 2.14

---

gtk_font_selection_dialog_get_ok_button ()

```
GtkWidget *
gtk_font_selection_dialog_get_ok_button
                                (GtkFontSelectionDialog *fsd);
```

`gtk_font_selection_dialog_get_ok_button` has been deprecated since version 3.2 and should not be used in newly-written code.

Use GtkFontChooserDialog

Gets the “OK” button.

Parameters

fsd	a GtkFontSelectionDialog

Returns

the GtkWidget used in the dialog for the “OK” button.

[transfer none]

Since: 2.14

---

gtk_font_selection_dialog_get_font_selection ()

```
GtkWidget *
gtk_font_selection_dialog_get_font_selection
                                (GtkFontSelectionDialog *fsd);
```

`gtk_font_selection_dialog_get_font_selection` has been deprecated since version 3.2 and should not be used in newly-written code.

Use GtkFontChooserDialog

Retrieves the GtkFontSelection widget embedded in the dialog.

Parameters

fsd	a GtkFontSelectionDialog

Returns

the embedded GtkFontSelection.

[transfer none]

Since: 2.22

Types and Values

struct GtkFontSelectionDialog

```
struct GtkFontSelectionDialog;
```

See Also

GtkFontSelection, GtkDialog, GtkFontChooserDialog

GtkHBox

GtkHBox — A horizontal container box

Functions

GtkWidget *	gtk_hbox_new ()

Types and Values

struct	GtkHBox

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkContainer
            └── GtkBox
                └── GtkHBox
```

Implemented Interfaces

GtkHBox implements AtkImplementorIface, GtkBuildable and GtkOrientable.

Includes

```
#include <gtk/gtk.h>
```

Description

GtkHBox is a container that organizes child widgets into a single row.

Use the GtkBox packing interface to determine the arrangement, spacing, width, and alignment of GtkHBox children.

All children are allocated the same height.

GtkHBox has been deprecated. You can use GtkBox instead, which is a very quick and easy change. If you have derived your own classes from GtkHBox, you can simply change the inheritance to derive directly from GtkBox. No further changes are needed, since the default value of the "orientation" property is `GTK_ORIENTATION_HORIZONTAL`. If you don't need first-child or last-child styling, and want your code to be future-proof, the recommendation is to switch to GtkGrid instead of nested boxes. For more information about migrating to GtkGrid, see Migrating from other containers to GtkGrid.

Functions

gtk_hbox_new ()

```
GtkWidget *
gtk_hbox_new (gboolean homogeneous,
              gint spacing);
```

`gtk_hbox_new` has been deprecated since version 3.2 and should not be used in newly-written code.

You can use `gtk_box_new()` with `GTK_ORIENTATION_HORIZONTAL` instead, which is a quick and easy change. But the recommendation is to switch to GtkGrid, since GtkBox is going to go away eventually. See Migrating from other containers to GtkGrid.

Creates a new GtkHBox.

Parameters

homogeneous	`TRUE` if all children are to be given equal space allotments.
spacing	the number of pixels to place by default between children.

Returns

a new GtkHBox.

Types and Values

struct GtkHBox

```
struct GtkHBox;
```

See Also

GtkVBox

GtkVBox

GtkVBox — A vertical container box

Functions

GtkWidget *	gtk_vbox_new ()

Types and Values

struct	GtkVBox

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkContainer
            └── GtkBox
                └── GtkVBox
```

Implemented Interfaces

GtkVBox implements AtkImplementorIface, GtkBuildable and GtkOrientable.

Includes

```
#include <gtk/gtk.h>
```

Description

A GtkVBox is a container that organizes child widgets into a single column.

Use the GtkBox packing interface to determine the arrangement, spacing, height, and alignment of GtkVBox children.

All children are allocated the same width.

GtkVBox has been deprecated. You can use GtkBox instead, which is a very quick and easy change. If you have derived your own classes from GtkVBox, you can simply change the inheritance to derive directly from GtkBox, and set the "orientation" property to `GTK_ORIENTATION_VERTICAL` in your instance init function, with a call like:

```
1 gtk_orientable_set_orientation
2 (GTK_ORIENTABLE (object),
  GTK_ORIENTATION_VERTICAL);
```

If you don't need first-child or last-child styling and want your code to be future-proof, the recommendation is to switch to GtkGrid instead of nested boxes. For more information about migrating to GtkGrid, see Migrating from other containers to GtkGrid.

Functions

gtk_vbox_new ()

```
GtkWidget *
gtk_vbox_new (gboolean homogeneous,
              gint spacing);
```

`gtk_vbox_new` has been deprecated since version 3.2 and should not be used in newly-written code.

You can use `gtk_box_new()` with `GTK_ORIENTATION_VERTICAL` instead, which is a quick and easy change. But the recommendation is to switch to GtkGrid, since GtkBox is going to go away eventually. See Migrating from other containers to GtkGrid.

Creates a new GtkVBox.

Parameters

homogeneous	`TRUE` if all children are to be given equal space allotments.
spacing	the number of pixels to place by default between children.

Returns

a new GtkVBox.

Types and Values

struct GtkVBox

```
struct GtkVBox;
```

See Also

GtkHBox

GtkHButtonBox

GtkHButtonBox — A container for arranging buttons horizontally

Functions

GtkWidget *	gtk_hbutton_box_new ()

Types and Values

struct	GtkHButtonBox

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkContainer
            └── GtkBox
                └── GtkButtonBox
                    └── GtkHButtonBox
```

Implemented Interfaces

GtkHButtonBox implements AtkImplementorIface, GtkBuildable and GtkOrientable.

Includes

```
#include <gtk/gtk.h>
```

Description

A button box should be used to provide a consistent layout of buttons throughout your application. The layout/spacing can be altered by the programmer, or if desired, by the user to alter the “feel” of a program to a small degree.

A GtkHButtonBox is created with `gtk_hbutton_box_new()`. Buttons are packed into a button box the same way widgets are added to any other container, using `gtk_container_add()`. You can also use `gtk_box_pack_start()` or `gtk_box_pack_end()`, but for button boxes both these functions work just like `gtk_container_add()`, ie., they pack the button in a way that depends on the current layout style and on whether the button has had `gtk_button_box_set_child_secondary()` called on it.

The spacing between buttons can be set with `gtk_box_set_spacing()`. The arrangement and layout of the buttons can be changed with `gtk_button_box_set_layout()`.

GtkHButtonBox has been deprecated, use GtkButtonBox instead.

Functions

gtk_hbutton_box_new ()

```
GtkWidget *
gtk_hbutton_box_new (void);
```

`gtk_hbutton_box_new` has been deprecated since version 3.2 and should not be used in newly-written code.

Use `gtk_button_box_new()` with `GTK_ORIENTATION_HORIZONTAL` instead

Creates a new horizontal button box.

Returns

a new button box GtkWidget.

Types and Values

struct GtkHButtonBox

```
struct GtkHButtonBox;
```

See Also

GtkBox, GtkButtonBox, GtkVButtonBox

GtkVButtonBox

GtkVButtonBox — A container for arranging buttons vertically

Functions

GtkWidget *	gtk_vbutton_box_new ()

Types and Values

struct	GtkVButtonBox

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkContainer
            └── GtkBox
                └── GtkButtonBox
                    └── GtkVButtonBox
```

Implemented Interfaces

GtkVButtonBox implements AtkImplementorIface, GtkBuildable and GtkOrientable.

Includes

```
#include <gtk/gtk.h>
```

Description

A button box should be used to provide a consistent layout of buttons throughout your application. The layout/spacing can be altered by the programmer, or if desired, by the user to alter the “feel” of a program to a small degree.

A GtkVButtonBox is created with `gtk_vbutton_box_new()`. Buttons are packed into a button box the same way widgets are added to any other container, using `gtk_container_add()`. You can also use `gtk_box_pack_start()` or `gtk_box_pack_end()`, but for button boxes both these functions work just like `gtk_container_add()`, ie., they pack the button in a way that depends on the current layout style and on whether the button has had `gtk_button_box_set_child_secondary()` called on it.

The spacing between buttons can be set with `gtk_box_set_spacing()`. The arrangement and layout of the buttons can be changed with `gtk_button_box_set_layout()`.

GtkVButtonBox has been deprecated, use GtkButtonBox instead.

Functions

gtk_vbutton_box_new ()

```
GtkWidget *
gtk_vbutton_box_new (void);
```

`gtk_vbutton_box_new` has been deprecated since version 3.2 and should not be used in newly-written code.

Use `gtk_button_box_new()` with `GTK_ORIENTATION_VERTICAL` instead

Creates a new vertical button box.

Returns

a new button box GtkWidget.

Types and Values

struct GtkVButtonBox

```
struct GtkVButtonBox;
```

See Also

GtkBox, GtkButtonBox, GtkHButtonBox

GtkHPaned

GtkHPaned — A container with two panes arranged horizontally

Functions

GtkWidget *	gtk_hpaned_new ()

Types and Values

struct	GtkHPaned

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkContainer
            └── GtkPaned
                └── GtkHPaned
```

Implemented Interfaces

GtkHPaned implements AtkImplementorIface, GtkBuildable and GtkOrientable.

Includes

```
#include <gtk/gtk.h>
```

Description

The HPaned widget is a container widget with two children arranged horizontally. The division between the two panes is adjustable by the user by dragging a handle. See GtkPaned for details.

GtkHPaned has been deprecated, use GtkPaned instead.

Functions

gtk_hpaned_new ()

```
GtkWidget *
gtk_hpaned_new (void);
```

`gtk_hpaned_new` has been deprecated since version 3.2 and should not be used in newly-written code.

Use `gtk_paned_new()` with `GTK_ORIENTATION_HORIZONTAL` instead

Create a new GtkHPaned

Returns

the new GtkHPaned

Types and Values

struct GtkHPaned

```
struct GtkHPaned;
```

GtkVPaned

GtkVPaned — A container with two panes arranged vertically

Functions

GtkWidget *	gtk_vpaned_new ()

Types and Values

struct	GtkVPaned

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkContainer
            └── GtkPaned
                └── GtkVPaned
```

Implemented Interfaces

GtkVPaned implements AtkImplementorIface, GtkBuildable and GtkOrientable.

Includes

```
#include <gtk/gtk.h>
```

Description

The VPaned widget is a container widget with two children arranged vertically. The division between the two

panes is adjustable by the user by dragging a handle. See GtkPaned for details.

GtkVPaned has been deprecated, use GtkPaned instead.

Functions

gtk_vpaned_new ()

```
GtkWidget *
gtk_vpaned_new (void);
```

`gtk_vpaned_new` has been deprecated since version 3.2 and should not be used in newly-written code.

Use `gtk_paned_new()` with `GTK_ORIENTATION_VERTICAL` instead

Create a new GtkVPaned

Returns

the new GtkVPaned

Types and Values

struct GtkVPaned

```
struct GtkVPaned;
```

GtkTable

GtkTable — Pack widgets in regular patterns

Functions

GtkWidget *	gtk_table_new ()
void	gtk_table_resize ()
void	gtk_table_get_size ()
void	gtk_table_attach ()
void	gtk_table_attach_defaults ()
void	gtk_table_set_row_spacing ()
void	gtk_table_set_col_spacing ()
void	gtk_table_set_row_spacings ()
void	gtk_table_set_col_spacings ()
void	gtk_table_set_homogeneous ()
guint	gtk_table_get_default_row_spacing ()
gboolean	gtk_table_get_homogeneous ()
guint	gtk_table_get_row_spacing ()

guint	gtk_table_get_col_spacing ()
guint	gtk_table_get_default_col_spacing ()

Properties

guint	column-spacing	Read / Write
gboolean	homogeneous	Read / Write
guint	n-columns	Read / Write
guint	n-rows	Read / Write
guint	row-spacing	Read / Write

Child Properties

guint	bottom-attach	Read / Write
guint	left-attach	Read / Write
guint	right-attach	Read / Write
guint	top-attach	Read / Write
GtkAttachOptions	x-options	Read / Write
guint	x-padding	Read / Write
GtkAttachOptions	y-options	Read / Write
guint	y-padding	Read / Write

Types and Values

struct	GtkTable
enum	GtkAttachOptions

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkContainer
            └── GtkTable
```

Implemented Interfaces

GtkTable implements AtkImplementorIface and GtkBuildable.

Includes

```
#include <gtk/gtk.h>
```

Description

The GtkTable functions allow the programmer to arrange widgets in rows and columns, making it easy to align many widgets next to each other, horizontally and vertically.

Tables are created with a call to `gtk_table_new()`, the size of which can later be changed with

gtk_table_resize().

Widgets can be added to a table using gtk_table_attach() or the more convenient (but slightly less flexible) gtk_table_attach_defaults().

To alter the space next to a specific row, use gtk_table_set_row_spacing(), and for a column, gtk_table_set_col_spacing(). The gaps between all rows or columns can be changed by calling gtk_table_set_row_spacings() or gtk_table_set_col_spacings() respectively. Note that spacing is added between the children, while padding added by gtk_table_attach() is added on either side of the widget it belongs to.

gtk_table_set_homogeneous(), can be used to set whether all cells in the table will resize themselves to the size of the largest widget in the table.

> GtkTable has been deprecated. Use GtkGrid instead. It provides the same capabilities as GtkTable for arranging widgets in a rectangular grid, but does support height-for-width geometry management.

Functions

gtk_table_new ()

```
GtkWidget *
gtk_table_new (guint rows,
               guint columns,
               gboolean homogeneous);
```

gtk_table_new has been deprecated since version 3.4 and should not be used in newly-written code.

Use gtk_grid_new().

Used to create a new table widget. An initial size must be given by specifying how many rows and columns the table should have, although this can be changed later with gtk_table_resize(). rows and columns must both be in the range 1 .. 65535. For historical reasons, 0 is accepted as well and is silently interpreted as 1.

Parameters

rows	The number of rows the new table should have.
columns	The number of columns the new table should have.
homogeneous	If set to TRUE, all table cells are resized to the size of the cell containing the largest widget.

Returns

A pointer to the newly created table widget.

---

gtk_table_resize ()

```
void
gtk_table_resize (GtkTable *table,
                  guint rows,
                  guint columns);
```

`gtk_table_resize` has been deprecated since version 3.4 and should not be used in newly-written code.

GtkGrid resizes automatically.

If you need to change a table's size after it has been created, this function allows you to do so.

Parameters

table	The GtkTable you wish to change the size of.	
rows	The new number of rows.	
columns	The new number of columns.	

---

gtk_table_get_size ()

```
void
gtk_table_get_size (GtkTable *table,
                    guint *rows,
                    guint *columns);
```

`gtk_table_get_size` has been deprecated since version 3.4 and should not be used in newly-written code.

GtkGrid does not expose the number of columns and rows.

Gets the number of rows and columns in the table.

Parameters

table	a GtkTable	
rows	return location for the number of rows, or `NULL`.	[out][allow-none]
columns	return location for the number of columns, or `NULL`.	[out][allow-none]

Since: 2.22

---

gtk_table_attach ()

```
void
gtk_table_attach (GtkTable *table,
                  GtkWidget *child,
                  guint left_attach,
                  guint right_attach,
                  guint top_attach,
                  guint bottom_attach,
                  GtkAttachOptions xoptions,
```

```
                    GtkAttachOptions yoptions,
                    guint xpadding,
                    guint ypadding);
```

`gtk_table_attach` has been deprecated since version 3.4 and should not be used in newly-written code.

Use `gtk_grid_attach()` with GtkGrid. Note that the attach arguments differ between those two functions.

Adds a widget to a table. The number of “cells” that a widget will occupy is specified by `left_attach` , `right_attach` , `top_attach` and `bottom_attach` . These each represent the leftmost, rightmost, uppermost and lowest column and row numbers of the table. (Columns and rows are indexed from zero).

To make a button occupy the lower right cell of a 2x2 table, use

```
gtk_table_attach (table, button,
                  1, 2, // left, right attach
                  1, 2, // top, bottom attach
                  xoptions, yoptions,
                  xpadding, ypadding);
```

If you want to make the button span the entire bottom row, use `left_attach` == 0 and `right_attach` = 2 instead.

Parameters

table	The GtkTable to add a new widget to.
child	The widget to add.
left_attach	the column number to attach the left side of a child widget to.
right_attach	the column number to attach the right side of a child widget to.
top_attach	the row number to attach the top of a child widget to.
bottom_attach	the row number to attach the bottom of a child widget to.
xoptions	Used to specify the properties of the child widget when the table is resized.
yoptions	The same as xoptions, except this field determines behaviour of vertical resizing.
xpadding	An integer value specifying the padding on the left and right of the widget being added to the table.
ypadding	The amount of padding above and below the child widget.

---

gtk_table_attach_defaults ()

```
void
gtk_table_attach_defaults (GtkTable *table,
                           GtkWidget *widget,
                           guint left_attach,
                           guint right_attach,
```

```
                            guint top_attach,
                            guint bottom_attach);
```

`gtk_table_attach_defaults` has been deprecated since version 3.4 and should not be used in newly-written code.

Use `gtk_grid_attach()` with GtkGrid. Note that the attach arguments differ between those two functions.

As there are many options associated with `gtk_table_attach()`, this convenience function provides the programmer with a means to add children to a table with identical padding and expansion options. The values used for the GtkAttachOptions are `GTK_EXPAND | GTK_FILL`, and the padding is set to 0.

Parameters

table	The table to add a new child widget to.
widget	The child widget to add.
left_attach	The column number to attach the left side of the child widget to.
right_attach	The column number to attach the right side of the child widget to.
top_attach	The row number to attach the top of the child widget to.
bottom_attach	The row number to attach the bottom of the child widget to.

---

gtk_table_set_row_spacing ()

```
void
gtk_table_set_row_spacing (GtkTable *table,
                           guint row,
                           guint spacing);
```

`gtk_table_set_row_spacing` has been deprecated since version 3.4 and should not be used in newly-written code.

Use `gtk_widget_set_margin_top()` and `gtk_widget_set_margin_bottom()` on the widgets contained in the row if you need this functionality. GtkGrid does not support per-row spacing.

Changes the space between a given table row and the subsequent row.

Parameters

table	a GtkTable containing the row whose properties you wish to change.
row	row number whose spacing will be changed.
spacing	number of pixels that the spacing should take up.

---

gtk_table_set_col_spacing ()

```
void
gtk_table_set_col_spacing (GtkTable *table,
                           guint column,
                           guint spacing);
```

`gtk_table_set_col_spacing` has been deprecated since version 3.4 and should not be used in newly-written code.

Use `gtk_widget_set_margin_start()` and `gtk_widget_set_margin_end()` on the widgets contained in the row if you need this functionality. GtkGrid does not support per-row spacing.

Alters the amount of space between a given table column and the following column.

Parameters

table	a GtkTable.
column	the column whose spacing should be changed.
spacing	number of pixels that the spacing should take up.

---

gtk_table_set_row_spacings ()

```
void
gtk_table_set_row_spacings (GtkTable *table,
                            guint spacing);
```

`gtk_table_set_row_spacings` has been deprecated since version 3.4 and should not be used in newly-written code.

Use `gtk_grid_set_row_spacing()` with GtkGrid.

Sets the space between every row in `table` equal to `spacing` .

Parameters

table	a GtkTable.
spacing	the number of pixels of space to place between every row in the table.

---

gtk_table_set_col_spacings ()

```
void
gtk_table_set_col_spacings (GtkTable *table,
                            guint spacing);
```

`gtk_table_set_col_spacings` has been deprecated since version 3.4 and should not be used in newly-written code.

Use `gtk_grid_set_column_spacing()` with GtkGrid.

Sets the space between every column in `table` equal to `spacing` .

Parameters

table	a GtkTable.
spacing	the number of pixels of space to place between every column in the table.

---

gtk_table_set_homogeneous ()

```
void
gtk_table_set_homogeneous (GtkTable *table,
                           gboolean homogeneous);
```

`gtk_table_set_homogeneous` has been deprecated since version 3.4 and should not be used in newly-written code.

Use `gtk_grid_set_row_homogeneous()` and `gtk_grid_set_column_homogeneous()` with GtkGrid.

Changes the homogenous property of table cells, ie. whether all cells are an equal size or not.

Parameters

table	The GtkTable you wish to set the homogeneous properties of.
homogeneous	Set to `TRUE` to ensure all table cells are the same size. Set to `FALSE` if this is not your desired behaviour.

---

gtk_table_get_default_row_spacing ()

```
guint
gtk_table_get_default_row_spacing (GtkTable *table);
```

`gtk_table_get_default_row_spacing` has been deprecated since version 3.4 and should not be used in newly-written code.

Use `gtk_grid_get_row_spacing()` with GtkGrid.

Gets the default row spacing for the table. This is the spacing that will be used for newly added rows. (See `gtk_table_set_row_spacings()`)

Parameters

table	a GtkTable

Returns

the default row spacing

---

gtk_table_get_homogeneous ()

```
gboolean
gtk_table_get_homogeneous (GtkTable *table);
```

`gtk_table_get_homogeneous` has been deprecated since version 3.4 and should not be used in newly-written code.

Use `gtk_grid_get_row_homogeneous()` and `gtk_grid_get_column_homogeneous()` with GtkGrid.

Returns whether the table cells are all constrained to the same width and height. (See `gtk_table_set_homogeneous()`)

Parameters

table	a GtkTable

Returns

`TRUE` if the cells are all constrained to the same size

---

gtk_table_get_row_spacing ()

```
guint
gtk_table_get_row_spacing (GtkTable *table,
                           guint row);
```

`gtk_table_get_row_spacing` has been deprecated since version 3.4 and should not be used in newly-written code.

GtkGrid does not offer a replacement for this functionality.

Gets the amount of space between row `row` , and row `row` + 1. See `gtk_table_set_row_spacing()`.

Parameters

table	a GtkTable
row	a row in the table, 0 indicates the first row

Returns

the row spacing

---

gtk_table_get_col_spacing ()

```
guint
gtk_table_get_col_spacing (GtkTable *table,
                           guint column);
```

`gtk_table_get_col_spacing` has been deprecated since version 3.4 and should not be used in newly-written code.

GtkGrid does not offer a replacement for this functionality.

Gets the amount of space between column `col` , and column `col` + 1. See `gtk_table_set_col_spacing()`.

Parameters

table	a GtkTable
column	a column in the table, 0 indicates the first column

Returns

the column spacing

---

gtk_table_get_default_col_spacing ()

```
guint
gtk_table_get_default_col_spacing (GtkTable *table);
```

`gtk_table_get_default_col_spacing` has been deprecated since version 3.4 and should not be used in newly-written code.

Use `gtk_grid_get_column_spacing()` with GtkGrid.

Gets the default column spacing for the table. This is the spacing that will be used for newly added columns. (See `gtk_table_set_col_spacings()`)

Parameters

table	a GtkTable

Returns

the default column spacing

Types and Values

struct GtkTable

```
struct GtkTable;
```

---

enum GtkAttachOptions

Denotes the expansion properties that a widget will have when it (or its parent) is resized.

Members

GTK_EXPAND	the widget should expand to take up any extra space in its container that has been allocated.
GTK_SHRINK	the widget should shrink as and when possible.
GTK_FILL	the widget should fill the space allocated to it.

Property Details

The `"column-spacing"` property

`"column-spacing"` `guint`

The amount of space between two consecutive columns.

Flags: Read / Write

Allowed values: <= 65535

Default value: 0

The `"homogeneous"` property

`"homogeneous"` `gboolean`

If TRUE, the table cells are all the same width/height.

Flags: Read / Write

Default value: FALSE

The `"n-columns"` property

`"n-columns"` `guint`

The number of columns in the table.

Flags: Read / Write

Allowed values: [1,65535]

Default value: 1

The “n-rows” property

“n-rows” guint

The number of rows in the table.

Flags: Read / Write

Allowed values: [1,65535]

Default value: 1

The “row-spacing” property

“row-spacing” guint

The amount of space between two consecutive rows.

Flags: Read / Write

Allowed values: <= 65535

Default value: 0

Child Property Details

The “bottom-attach” child property

“bottom-attach” guint

The row number to attach the bottom of the child to.

Flags: Read / Write

Allowed values: [1,65535]

Default value: 1

The “left-attach” child property

“left-attach” guint

The column number to attach the left side of the child to.

Flags: Read / Write

Allowed values: <= 65535

Default value: 0

The “right-attach” child property

“right-attach” guint

The column number to attach the right side of a child widget to.

Flags: Read / Write

Allowed values: [1,65535]

Default value: 1

The “top-attach” child property

“top-attach” guint

The row number to attach the top of a child widget to.

Flags: Read / Write

Allowed values: <= 65535

Default value: 0

The “x-options” child property

“x-options” GtkAttachOptions

Options specifying the horizontal behaviour of the child.

Flags: Read / Write

Default value: GTK_EXPAND | GTK_FILL

The “x-padding” child property

“x-padding” guint

Extra space to put between the child and its left and right neighbors, in pixels.

Flags: Read / Write

Allowed values: <= 65535

Default value: 0

The “y-options” child property

“y-options” GtkAttachOptions

Options specifying the vertical behaviour of the child.

Flags: Read / Write

Default value: GTK_EXPAND | GTK_FILL

The “y-padding” child property

“y-padding” guint

Extra space to put between the child and its upper and lower neighbors, in pixels.

Flags: Read / Write

Allowed values: <= 65535

Default value: 0

See Also

GtkGrid

GtkHSeparator

GtkHSeparator — A horizontal separator

Functions

GtkWidget *	gtk_hseparator_new ()

Types and Values

struct	GtkHSeparator

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkSeparator
            └── GtkHSeparator
```

Implemented Interfaces

GtkHSeparator implements AtkImplementorIface, GtkBuildable and GtkOrientable.

Includes

```
#include <gtk/gtk.h>
```

Description

The GtkHSeparator widget is a horizontal separator, used to group the widgets within a window. It displays a horizontal line with a shadow to make it appear sunken into the interface.

> The GtkHSeparator widget is not used as a separator within menus. To create a separator in a menu create an empty GtkSeparatorMenuItem widget using `gtk_separator_menu_item_new()` and add it to the menu with `gtk_menu_shell_append()`.

GtkHSeparator has been deprecated, use GtkSeparator instead.

Functions

gtk_hseparator_new ()

```
GtkWidget *
gtk_hseparator_new (void);
```

`gtk_hseparator_new` has been deprecated since version 3.2 and should not be used in newly-written code.

Use `gtk_separator_new()` with `GTK_ORIENTATION_HORIZONTAL` instead

Creates a new GtkHSeparator.

Returns

a new GtkHSeparator.

Types and Values

struct GtkHSeparator

```
struct GtkHSeparator;
```

See Also

GtkSeparator

GtkVSeparator

GtkVSeparator — A vertical separator

Functions

GtkWidget *	gtk_vseparator_new ()

Types and Values

struct	GtkVSeparator

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkSeparator
            └── GtkVSeparator
```

Implemented Interfaces

GtkVSeparator implements AtkImplementorIface, GtkBuildable and GtkOrientable.

Includes

```
#include <gtk/gtk.h>
```

Description

The GtkVSeparator widget is a vertical separator, used to group the widgets within a window. It displays a vertical line with a shadow to make it appear sunken into the interface.

GtkVSeparator has been deprecated, use GtkSeparator instead.

Functions

gtk_vseparator_new ()

```
GtkWidget *
gtk_vseparator_new (void);
```

`gtk_vseparator_new` has been deprecated since version 3.2 and should not be used in newly-written code.

Use `gtk_separator_new()` with `GTK_ORIENTATION_VERTICAL` instead

Creates a new GtkVSeparator.

Returns

a new GtkVSeparator.

Types and Values

struct GtkVSeparator

```
struct GtkVSeparator;
```

The GtkVSeparator struct contains private data only, and should be accessed using the functions below.

See Also

GtkHSeparator

GtkHScrollbar

GtkHScrollbar — A horizontal scrollbar

Functions

GtkWidget *	gtk_hscrollbar_new ()

Types and Values

struct	GtkHScrollbar

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkRange
            └── GtkScrollbar
                └── GtkHScrollbar
```

Implemented Interfaces

GtkHScrollbar implements AtkImplementorIface, GtkBuildable and GtkOrientable.

Includes

```
#include <gtk/gtk.h>
```

Description

The GtkHScrollbar widget is a widget arranged horizontally creating a scrollbar. See GtkScrollbar for details on scrollbars. GtkAdjustment pointers may be added to handle the adjustment of the scrollbar or it may be left `NULL` in which case one will be created for you. See GtkScrollbar for a description of what the fields in an adjustment represent for a scrollbar.

GtkHScrollbar has been deprecated, use GtkScrollbar instead.

Functions

gtk_hscrollbar_new ()

```
GtkWidget *
gtk_hscrollbar_new (GtkAdjustment *adjustment);
```

`gtk_hscrollbar_new` has been deprecated since version 3.2 and should not be used in newly-written code.

Use `gtk_scrollbar_new()` with `GTK_ORIENTATION_HORIZONTAL` instead

Creates a new horizontal scrollbar.

Parameters

adjustment	the GtkAdjustment to use, or `NULL` to create a new adjustment.	[allow-none]

Returns

the new GtkHScrollbar

Types and Values

struct GtkHScrollbar

```
struct GtkHScrollbar;
```

See Also

GtkScrollbar, GtkScrolledWindow

GtkVScrollbar

GtkVScrollbar — A vertical scrollbar

Functions

GtkWidget *	gtk_vscrollbar_new ()

Types and Values

struct	GtkVScrollbar

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkRange
            └── GtkScrollbar
                └── GtkVScrollbar
```

Implemented Interfaces

GtkVScrollbar implements AtkImplementorIface, GtkBuildable and GtkOrientable.

Includes

```
#include <gtk/gtk.h>
```

Description

The GtkVScrollbar widget is a widget arranged vertically creating a scrollbar. See GtkScrollbar for details on scrollbars. GtkAdjustment pointers may be added to handle the adjustment of the scrollbar or it may be left `NULL` in which case one will be created for you. See GtkScrollbar for a description of what the fields in an adjustment represent for a scrollbar.

GtkVScrollbar has been deprecated, use GtkScrollbar instead.

Functions

gtk_vscrollbar_new ()

```
GtkWidget *
gtk_vscrollbar_new (GtkAdjustment *adjustment);
```

`gtk_vscrollbar_new` has been deprecated since version 3.2 and should not be used in newly-written code.

Use `gtk_scrollbar_new()` with `GTK_ORIENTATION_VERTICAL` instead

Creates a new vertical scrollbar.

Parameters

adjustment	the GtkAdjustment to use, or `NULL` to create a new adjustment.	[allow-none]

Returns

the new GtkVScrollbar

Types and Values

struct GtkVScrollbar

```
struct GtkVScrollbar;
```

The GtkVScrollbar struct contains private data and should be accessed using the functions below.

See Also

GtkScrollbar, GtkScrolledWindow

GtkUIManager

GtkUIManager — Constructing menus and toolbars from an XML description

Functions

GtkUIManager *	gtk_ui_manager_new ()
void	gtk_ui_manager_set_add_tearoffs ()
gboolean	gtk_ui_manager_get_add_tearoffs ()
void	gtk_ui_manager_insert_action_group ()
void	gtk_ui_manager_remove_action_group ()
GList *	gtk_ui_manager_get_action_groups ()
GtkAccelGroup *	gtk_ui_manager_get_accel_group ()
GtkWidget *	gtk_ui_manager_get_widget ()
GSList *	gtk_ui_manager_get_toplevels ()
GtkAction *	gtk_ui_manager_get_action ()
guint	gtk_ui_manager_add_ui_from_resource ()
guint	gtk_ui_manager_add_ui_from_string ()
guint	gtk_ui_manager_add_ui_from_file ()
guint	gtk_ui_manager_new_merge_id ()
void	gtk_ui_manager_add_ui ()
void	gtk_ui_manager_remove_ui ()
gchar *	gtk_ui_manager_get_ui ()
void	gtk_ui_manager_ensure_update ()

Properties

gboolean	add-tearoffs	Read / Write
gchar *	ui	Read

Signals

void	actions-changed	No Recursion
void	add-widget	No Recursion

void	connect-proxy	No Recursion
void	disconnect-proxy	No Recursion
void	post-activate	No Recursion
void	pre-activate	No Recursion

Types and Values

struct	GtkUIManager
enum	GtkUIManagerItemType

Object Hierarchy

```
GObject
└── GtkUIManager
```

Implemented Interfaces

GtkUIManager implements GtkBuildable.

Includes

```
#include <gtk/gtk.h>
```

Description

> GtkUIManager is deprecated since GTK+ 3.10. To construct user interfaces from XML definitions, you should use GtkBuilder, GMenuModel, et al. To work with actions, use GAction, GtkActionable et al. These newer classes support richer functionality and integration with various desktop shells. It should be possible to migrate most/all functionality from GtkUIManager.

A GtkUIManager constructs a user interface (menus and toolbars) from one or more UI definitions, which reference actions from one or more action groups.

UI Definitions

The UI definitions are specified in an XML format which can be roughly described by the following DTD.

> Do not confuse the GtkUIManager UI Definitions described here with the similarly named GtkBuilder UI Definitions.

```
<!ELEMENT ui          (menubar|toolbar|popup|
accelerator)* >
<!ELEMENT menubar     (menuitem|separator|
placeholder|menu)* >
<!ELEMENT menu        (menuitem|separator|
placeholder|menu)* >
<!ELEMENT popup       (menuitem|separator|
placeholder|menu)* >
<!ELEMENT toolbar     (toolitem|separator|
placeholder)* >
<!ELEMENT placeholder (menuitem|toolitem|
```

```
separator|placeholder|menu)* >
<!ELEMENT menuitem     EMPTY >
<!ELEMENT toolitem     (menu?) >
<!ELEMENT separator    EMPTY >
<!ELEMENT accelerator  EMPTY >
<!ATTLIST menubar      name
#IMPLIED
                       action
#IMPLIED >
<!ATTLIST toolbar      name
#IMPLIED
                       action
#IMPLIED >
<!ATTLIST popup        name
#IMPLIED
                       action
#IMPLIED
                       accelerators (true|
false) #IMPLIED >
<!ATTLIST placeholder  name
#IMPLIED
                       action
#IMPLIED >
<!ATTLIST separator    name
#IMPLIED
                       action
#IMPLIED
                       expand       (true|
false) #IMPLIED >
<!ATTLIST menu         name
#IMPLIED
                       action
#REQUIRED
                       position     (top|bot)
#IMPLIED >
<!ATTLIST menuitem     name
#IMPLIED
                       action
#REQUIRED
                       position     (top|bot)
#IMPLIED
                       always-show-image
(true|false) #IMPLIED >
<!ATTLIST toolitem     name
#IMPLIED
                       action
#REQUIRED
                       position     (top|bot)
#IMPLIED >
<!ATTLIST accelerator  name
#IMPLIED
                       action
#REQUIRED >
```

There are some additional restrictions beyond those specified in the DTD, e.g. every toolitem must have a toolbar in its anchestry and every menuitem must have a menubar or popup in its anchestry. Since a GMarkupParser is used to parse the UI description, it must not only be valid XML, but valid markup.

If a name is not specified, it defaults to the action. If an action is not specified either, the element name is used. The name and action attributes must not contain “/” characters after parsing (since that would mess up path lookup) and must be usable as XML attributes when enclosed in doublequotes, thus they must not “"” characters or references to the " entity.

A UI definition

```
<ui>
  <menubar>
    <menu name="FileMenu"
action="FileMenuAction">
      <menuitem name="New"
action="New2Action" />
      <placeholder name="FileMenuAdditions" /
>
    </menu>
    <menu name="JustifyMenu"
action="JustifyMenuAction">
      <menuitem name="Left" action="justify-
left"/>
      <menuitem name="Centre"
action="justify-center"/>
      <menuitem name="Right" action="justify-
right"/>
      <menuitem name="Fill" action="justify-
fill"/>
    </menu>
  </menubar>
  <toolbar action="toolbar1">
    <placeholder name="JustifyToolItems">
      <separator/>
      <toolitem name="Left" action="justify-
left"/>
      <toolitem name="Centre"
action="justify-center"/>
      <toolitem name="Right" action="justify-
right"/>
      <toolitem name="Fill" action="justify-
fill"/>
      <separator/>
    </placeholder>
  </toolbar>
</ui>
```

The constructed widget hierarchy is very similar to the element tree of the XML, with the exception that placeholders are merged into their parents. The correspondence of XML elements to widgets should be almost obvious:

- menubar

  a GtkMenuBar

- toolbar

  a GtkToolbar

- popup

  a toplevel GtkMenu

- menu

  a GtkMenu attached to a menuitem

- menuitem

  a GtkMenuItem subclass, the exact type depends on the action

- toolitem

  a GtkToolItem subclass, the exact type depends on the action. Note that toolitem elements may contain a menu element, but only if their associated action specifies a GtkMenuToolButton as proxy.

- separator

  a GtkSeparatorMenuItem or GtkSeparatorToolItem

- accelerator

  a keyboard accelerator

The “position” attribute determines where a constructed widget is positioned wrt. to its siblings in the partially constructed tree. If it is “top”, the widget is prepended, otherwise it is appended.

UI Merging

The most remarkable feature of GtkUIManager is that it can overlay a set of menuitems and toolitems over another one, and demerge them later.

Merging is done based on the names of the XML elements. Each element is identified by a path which consists of the names of its anchestors, separated by slashes. For example, the menuitem named “Left” in the example above has the path `/ui/menubar/JustifyMenu/Left` and the toolitem with the same name has path `/ui/toolbar1/JustifyToolItems/Left`.

Accelerators

Every action has an accelerator path. Accelerators are installed together with menuitem proxies, but they can also be explicitly added with <accelerator> elements in the UI definition. This makes it possible to have accelerators for actions even if they have no visible proxies.

Smart Separators

The separators created by GtkUIManager are “smart”, i.e. they do not show up in the UI unless they end up between two visible menu or tool items. Separators which are located at the very beginning or end of the menu or toolbar containing them, or multiple separators next to each other, are hidden. This is a useful feature, since the merging of UI elements from multiple sources can make it hard or impossible to determine in advance whether a separator will end up in such an unfortunate position.

For separators in toolbars, you can set `expand="true"` to turn them from a small, visible separator to an expanding, invisible one. Toolitems following an expanding separator are effectively right-aligned.

Empty Menus

Submenus pose similar problems to separators inconnection with merging. It is impossible to know in advance whether they will end up empty after merging. GtkUIManager offers two ways to treat empty submenus:

- make them disappear by hiding the menu item they’re attached to
- add an insensitive “Empty” item

The behaviour is chosen based on the “hide_if_empty” property of the action to which the submenu is associated.

GtkUIManager as GtkBuildable

The GtkUIManager implementation of the GtkBuildable interface accepts GtkActionGroup objects as <child> elements in UI definitions.

A GtkUIManager UI definition as described above can be embedded in an GtkUIManager <object> element in a GtkBuilder UI definition.

The widgets that are constructed by a GtkUIManager can be embedded in other parts of the constructed user interface with the help of the “constructor” attribute. See the example below.

An embedded GtkUIManager UI definition

```
<object class="GtkUIManager" id="uiman">
  <child>
    <object class="GtkActionGroup"
id="actiongroup">
      <child>
        <object class="GtkAction" id="file">
          <property
name="label">_File</property>
        </object>
      </child>
    </object>
  </child>
  <ui>
    <menubar name="menubar1">
      <menu action="file">
      </menu>
    </menubar>
  </ui>
</object>
<object class="GtkWindow" id="main-window">
  <child>
    <object class="GtkMenuBar" id="menubar1"
constructor="uiman"/>
  </child>
</object>
```

Functions

gtk_ui_manager_new ()

```
GtkUIManager *
gtk_ui_manager_new (void);
```

`gtk_ui_manager_new` has been deprecated since version 3.10 and should not be used in newly-written code.

Creates a new ui manager object.

Returns

a new ui manager object.

Since: 2.4

---

gtk_ui_manager_set_add_tearoffs ()

```
void
gtk_ui_manager_set_add_tearoffs (GtkUIManager *manager,
                                 gboolean add_tearoffs);
```

`gtk_ui_manager_set_add_tearoffs` has been deprecated since version 3.4 and should not be used in newly-written code.

Tearoff menus are deprecated and should not be used in newly written code.

Sets the “add_tearoffs” property, which controls whether menus generated by this GtkUIManager will have tearoff menu items.

Note that this only affects regular menus. Generated popup menus never have tearoff menu items.

Parameters

manager	a GtkUIManager
add_tearoffs	whether tearoff menu items are added

Since: 2.4

---

gtk_ui_manager_get_add_tearoffs ()

```
gboolean
gtk_ui_manager_get_add_tearoffs (GtkUIManager *manager);
```

`gtk_ui_manager_get_add_tearoffs` has been deprecated since version 3.4 and should not be used in newly-written code.

Tearoff menus are deprecated and should not be used in newly written code.

Returns whether menus generated by this GtkUIManager will have tearoff menu items.

Parameters

manager	a GtkUIManager

Returns

whether tearoff menu items are added

Since: 2.4

---

gtk_ui_manager_insert_action_group ()

```
void
gtk_ui_manager_insert_action_group (GtkUIManager *manager,
                                    GtkActionGroup *action_group,
                                    gint pos);
```

`gtk_ui_manager_insert_action_group` has been deprecated since version 3.10 and should not be used in newly-written code.

Inserts an action group into the list of action groups associated with `manager` . Actions in earlier groups hide actions with the same name in later groups.

If `pos` is larger than the number of action groups in `manager` , or negative, `action_group` will be inserted at the end of the internal list.

Parameters

manager	a GtkUIManager object
action_group	the action group to be inserted
pos	the position at which the group will be inserted.

Since: 2.4

---

gtk_ui_manager_remove_action_group ()

```
void
gtk_ui_manager_remove_action_group (GtkUIManager *manager,
                                    GtkActionGroup *action_group);
```

`gtk_ui_manager_remove_action_group` has been deprecated since version 3.10 and should not be used in newly-written code.

Removes an action group from the list of action groups associated with `manager` .

Parameters

manager	a GtkUIManager object
action_group	the action group to be removed

Since: 2.4

---

gtk_ui_manager_get_action_groups ()

```
GList *
gtk_ui_manager_get_action_groups (GtkUIManager *manager);
```

`gtk_ui_manager_get_action_groups` has been deprecated since version 3.10 and should not be used in newly-written code.

Returns the list of action groups associated with `manager` .

Parameters

manager	a GtkUIManager object

Returns

a GList of action groups. The list is owned by GTK+ and should not be modified.

[element-type GtkActionGroup][transfer none]

Since: 2.4

---

gtk_ui_manager_get_accel_group ()

```
GtkAccelGroup *
gtk_ui_manager_get_accel_group (GtkUIManager *manager);
```

`gtk_ui_manager_get_accel_group` has been deprecated since version 3.10 and should not be used in newly-written code.

Returns the GtkAccelGroup associated with `manager` .

Parameters

manager	a GtkUIManager object

Returns

the GtkAccelGroup.

[transfer none]

Since: 2.4

---

gtk_ui_manager_get_widget ()

```
GtkWidget *
gtk_ui_manager_get_widget (GtkUIManager *manager,
                           const gchar *path);
```

`gtk_ui_manager_get_widget` has been deprecated since version 3.10 and should not be used in newly-written code.

Looks up a widget by following a path. The path consists of the names specified in the XML description of the UI. separated by “/”. Elements which don’t have a name or action attribute in the XML (e.g. <popup>) can be addressed by their XML element name (e.g. "popup"). The root element ("/ui") can be omitted in the path.

Note that the widget found by following a path that ends in a <menu>; element is the menuitem to which the menu is attached, not the menu it manages.

Also note that the widgets constructed by a ui manager are not tied to the lifecycle of the ui manager. If you add the widgets returned by this function to some container or explicitly ref them, they will survive the destruction of the ui manager.

Parameters

manager	a GtkUIManager
path	a path

Returns

the widget found by following the path, or `NULL` if no widget was found.

[transfer none]

Since: 2.4

---

gtk_ui_manager_get_toplevels ()

```
GSList *
gtk_ui_manager_get_toplevels (GtkUIManager *manager,
                              GtkUIManagerItemType types);
```

`gtk_ui_manager_get_toplevels` has been deprecated since version 3.10 and should not be used in newly-written code.

Obtains a list of all toplevel widgets of the requested types.

Parameters

manager	a GtkUIManager
types	specifies the types of toplevel widgets to include. Allowed types are GTK_UI_MANAGER_MENUBAR, GTK_UI_MANAGER_TOOLBAR and GTK_UI_MANAGER_POPUP.

Returns

a newly-allocated GSList of all toplevel widgets of the requested types. Free the returned list with

`g_slist_free()`.

[element-type GtkWidget][transfer container]

Since: 2.4

---

gtk_ui_manager_get_action ()

```
GtkAction *
gtk_ui_manager_get_action (GtkUIManager *manager,
                           const gchar *path);
```

`gtk_ui_manager_get_action` has been deprecated since version 3.10 and should not be used in newly-written code.

Looks up an action by following a path. See `gtk_ui_manager_get_widget()` for more information about paths.

Parameters

manager	a GtkUIManager
path	a path

Returns

the action whose proxy widget is found by following the path, or `NULL` if no widget was found.

[transfer none]

Since: 2.4

---

gtk_ui_manager_add_ui_from_resource ()

```
guint
gtk_ui_manager_add_ui_from_resource (GtkUIManager *manager,
                                     const gchar *resource_path,
                                     GError **error);
```

`gtk_ui_manager_add_ui_from_resource` has been deprecated since version 3.10 and should not be used in newly-written code.

Parses a resource file containing a UI definition and merges it with the current contents of `manager` .

Parameters

manager	a GtkUIManager object
resource_path	the resource path of the file to parse
error	return location for an error

Returns

The merge id for the merged UI. The merge id can be used to unmerge the UI with `gtk_ui_manager_remove_ui()`. If an error occurred, the return value is 0.

Since: 3.4

---

gtk_ui_manager_add_ui_from_string ()

```
guint
gtk_ui_manager_add_ui_from_string (GtkUIManager *manager,
                                   const gchar *buffer,
                                   gssize length,
                                   GError **error);
```

`gtk_ui_manager_add_ui_from_string` has been deprecated since version 3.10 and should not be used in newly-written code.

Parses a string containing a UI definition and merges it with the current contents of `manager` . An enclosing <ui> element is added if it is missing.

Parameters

manager	a GtkUIManager object
buffer	the string to parse
length	the length of `buffer` (may be -1 if `buffer` is nul-terminated)
error	return location for an error

Returns

The merge id for the merged UI. The merge id can be used to unmerge the UI with `gtk_ui_manager_remove_ui()`. If an error occurred, the return value is 0.

Since: 2.4

---

gtk_ui_manager_add_ui_from_file ()

```
guint
gtk_ui_manager_add_ui_from_file (GtkUIManager *manager,
                                 const gchar *filename,
                                 GError **error);
```

`gtk_ui_manager_add_ui_from_file` has been deprecated since version 3.10 and should not be used in newly-written code.

Parses a file containing a UI definition and merges it with the current contents of `manager` .

Parameters

manager	a GtkUIManager object

filename	the name of the file to parse.	[type filename]
error	return location for an error	

Returns

The merge id for the merged UI. The merge id can be used to unmerge the UI with `gtk_ui_manager_remove_ui()`. If an error occurred, the return value is 0.

Since: 2.4

---

gtk_ui_manager_new_merge_id ()

```
guint
gtk_ui_manager_new_merge_id (GtkUIManager *manager);
```

`gtk_ui_manager_new_merge_id` has been deprecated since version 3.10 and should not be used in newly-written code.

Returns an unused merge id, suitable for use with `gtk_ui_manager_add_ui()`.

Parameters

manager	a GtkUIManager

Returns

an unused merge id.

Since: 2.4

---

gtk_ui_manager_add_ui ()

```
void
gtk_ui_manager_add_ui (GtkUIManager *manager,
                       guint merge_id,
                       const gchar *path,
                       const gchar *name,
                       const gchar *action,
                       GtkUIManagerItemType type,
                       gboolean top);
```

`gtk_ui_manager_add_ui` has been deprecated since version 3.10 and should not be used in newly-written code.

Adds a UI element to the current contents of `manager` .

If `type` is `GTK_UI_MANAGER_AUTO`, GTK+ inserts a menuitem, toolitem or separator if such an element can be inserted at the place determined by `path` . Otherwise `type` must indicate an element that can be inserted at the place determined by `path` .

If `path` points to a menuitem or toolitem, the new element will be inserted before or after this item, depending on `top` .

Parameters

manager	a GtkUIManager	
merge_id	the merge id for the merged UI, see gtk_ui_manager_new_merge_id()	
path	a path	
name	the name for the added UI element	
action	the name of the action to be proxied, or NULL to add a separator.	[allow-none]
type	the type of UI element to add.	
top	if TRUE, the UI element is added before its siblings, otherwise it is added after its siblings.	

Since: 2.4

---

gtk_ui_manager_remove_ui ()

```
void
gtk_ui_manager_remove_ui (GtkUIManager *manager,
                          guint merge_id);
```

gtk_ui_manager_remove_ui has been deprecated since version 3.10 and should not be used in newly-written code.

Unmerges the part of manager 's content identified by merge_id .

Parameters

manager	a GtkUIManager object
merge_id	a merge id as returned by gtk_ui_manager_add_ui_from_string()

Since: 2.4

---

gtk_ui_manager_get_ui ()

```
gchar *
gtk_ui_manager_get_ui (GtkUIManager *manager);
```

gtk_ui_manager_get_ui has been deprecated since version 3.10 and should not be used in newly-written code.

Creates a UI definition of the merged UI.

Parameters

manager	a GtkUIManager

Returns

A newly allocated string containing an XML representation of the merged UI.

Since: 2.4

---

gtk_ui_manager_ensure_update ()

```
void
gtk_ui_manager_ensure_update (GtkUIManager *manager);
```

`gtk_ui_manager_ensure_update` has been deprecated since version 3.10 and should not be used in newly-written code.

Makes sure that all pending updates to the UI have been completed.

This may occasionally be necessary, since GtkUIManager updates the UI in an idle function. A typical example where this function is useful is to enforce that the menubar and toolbar have been added to the main window before showing it:

```
gtk_container_add (GTK_CONTAINER (window), vbox);
g_signal_connect (merge, "add-widget",
                  G_CALLBACK (add_widget), vbox);
gtk_ui_manager_add_ui_from_file (merge, "my-menus");
gtk_ui_manager_add_ui_from_file (merge, "my-toolbars");
gtk_ui_manager_ensure_update (merge);
gtk_widget_show (window);
```

Parameters

manager	a GtkUIManager

Since: 2.4

Types and Values

struct GtkUIManager

```
struct GtkUIManager;
```

---

enum GtkUIManagerItemType

`GtkUIManagerItemType` has been deprecated since version 3.10 and should not be used in newly-written code.

These enumeration values are used by `gtk_ui_manager_add_ui()` to determine what UI element to create.

Members

GTK_UI_MANAGER_AUTO	Pick the type of the UI element according to context.
GTK_UI_MANAGER_MENUBAR	Create a menubar.
GTK_UI_MANAGER_MENU	Create a menu.
GTK_UI_MANAGER_TOOLBAR	Create a toolbar.
GTK_UI_MANAGER_PLACEHOLDER	Insert a placeholder.
GTK_UI_MANAGER_POPUP	Create a popup menu.
GTK_UI_MANAGER_MENUITEM	Create a menuitem.
GTK_UI_MANAGER_TOOLITEM	Create a toolitem.
GTK_UI_MANAGER_SEPARATOR	Create a separator.
GTK_UI_MANAGER_ACCELERATOR	Install an accelerator.
GTK_UI_MANAGER_POPUP_WITH_ACCELS	Same as GTK_UI_MANAGER_POPUP, but the actions' accelerators are shown.

Property Details

The "add-tearoffs" property

```
  "add-tearoffs"             gboolean
```

The "add-tearoffs" property controls whether generated menus have tearoff menu items.

Note that this only affects regular menus. Generated popup menus never have tearoff menu items.

GtkUIManager:add-tearoffs has been deprecated since version 3.4 and should not be used in newly-written code.

Tearoff menus are deprecated and should not be used in newly written code.

Flags: Read / Write

Default value: FALSE

Since: 2.4

The "ui" property

```
  "ui"                       gchar *
```

An XML string describing the merged UI.

Flags: Read

Default value: "<ui>\n</ui>\n"

Signal Details

The `"actions-changed"` signal

```
void
user_function (GtkUIManager *manager,
               gpointer      user_data)
```

The ::actions-changed signal is emitted whenever the set of actions changes.

`GtkUIManager::actions-changed` has been deprecated since version 3.10 and should not be used in newly-written code.

Parameters

manager	a GtkUIManager
user_data	user data set when the signal handler was connected.

Flags: No Recursion

Since: 2.4

The `"add-widget"` signal

```
void
user_function (GtkUIManager *manager,
               GtkWidget    *widget,
               gpointer      user_data)
```

The ::add-widget signal is emitted for each generated menubar and toolbar. It is not emitted for generated popup menus, which can be obtained by `gtk_ui_manager_get_widget()`.

`GtkUIManager::add-widget` has been deprecated since version 3.10 and should not be used in newly-written code.

Parameters

manager	a GtkUIManager
widget	the added widget
user_data	user data set when the signal handler was connected.

Flags: No Recursion

Since: 2.4

The “connect-proxy” signal

```
void
user_function (GtkUIManager *manager,
               GtkAction    *action,
               GtkWidget    *proxy,
               gpointer      user_data)
```

The ::connect-proxy signal is emitted after connecting a proxy to an action in the group.

This is intended for simple customizations for which a custom action class would be too clumsy, e.g. showing tooltips for menuitems in the statusbar.

`GtkUIManager::connect-proxy` has been deprecated since version 3.10 and should not be used in newly-written code.

Parameters

manager	the ui manager
action	the action
proxy	the proxy
user_data	user data set when the signal handler was connected.

Flags: No Recursion

Since: 2.4

The “disconnect-proxy” signal

```
void
user_function (GtkUIManager *manager,
               GtkAction    *action,
               GtkWidget    *proxy,
               gpointer      user_data)
```

The ::disconnect-proxy signal is emitted after disconnecting a proxy from an action in the group.

`GtkUIManager::disconnect-proxy` has been deprecated since version 3.10 and should not be used in newly-written code.

Parameters

manager	the ui manager
action	the action
proxy	the proxy
user_data	user data set when the signal handler was connected.

Flags: No Recursion

Since: 2.4

The “post-activate” signal

```
void
user_function (GtkUIManager *manager,
               GtkAction    *action,
               gpointer      user_data)
```

The ::post-activate signal is emitted just after the `action` is activated.

This is intended for applications to get notification just after any action is activated.

`GtkUIManager::post-activate` has been deprecated since version 3.10 and should not be used in newly-written code.

Parameters

manager	the ui manager
action	the action
user_data	user data set when the signal handler was connected.

Flags: No Recursion

Since: 2.4

The “pre-activate” signal

```
void
user_function (GtkUIManager *manager,
               GtkAction    *action,
               gpointer      user_data)
```

The ::pre-activate signal is emitted just before the `action` is activated.

This is intended for applications to get notification just before any action is activated.

`GtkUIManager::pre-activate` has been deprecated since version 3.10 and should not be used in newly-written code.

Parameters

manager	the ui manager
action	the action
user_data	user data set when the signal handler was connected.

Flags: No Recursion

Since: 2.4

See Also

GtkBuilder

GtkActionGroup

GtkActionGroup — A group of actions

Functions

GtkActionGroup *	gtk_action_group_new ()
const gchar *	gtk_action_group_get_name ()
gboolean	gtk_action_group_get_sensitive ()
void	gtk_action_group_set_sensitive ()
gboolean	gtk_action_group_get_visible ()
void	gtk_action_group_set_visible ()
GtkAccelGroup *	gtk_action_group_get_accel_group ()
void	gtk_action_group_set_accel_group ()
GtkAction *	gtk_action_group_get_action ()
GList *	gtk_action_group_list_actions ()
void	gtk_action_group_add_action ()
void	gtk_action_group_add_action_with_accel ()
void	gtk_action_group_remove_action ()
void	gtk_action_group_add_actions ()
void	gtk_action_group_add_actions_full ()
void	gtk_action_group_add_toggle_actions ()
void	gtk_action_group_add_toggle_actions_full ()
void	gtk_action_group_add_radio_actions ()
void	gtk_action_group_add_radio_actions_full ()
gchar *	(*GtkTranslateFunc) ()
void	gtk_action_group_set_translate_func ()
void	gtk_action_group_set_translation_domain ()
const gchar *	gtk_action_group_translate_string ()

Properties

GtkAccelGroup *	accel-group	Read / Write
gchar *	name	Read / Write / Construct Only
gboolean	sensitive	Read / Write
gboolean	visible	Read / Write

Signals

void	connect-proxy
void	disconnect-proxy
void	post-activate
void	pre-activate

Types and Values

struct	GtkActionGroup
struct	GtkActionGroupClass

struct	GtkActionEntry
struct	GtkToggleActionEntry
struct	GtkRadioActionEntry

Object Hierarchy

```
GObject
└── GtkActionGroup
```

Implemented Interfaces

GtkActionGroup implements GtkBuildable.

Includes

```
#include <gtk/gtk.h>
```

Description

Actions are organised into groups. An action group is essentially a map from names to GtkAction objects.

All actions that would make sense to use in a particular context should be in a single group. Multiple action groups may be used for a particular user interface. In fact, it is expected that most nontrivial applications will make use of multiple groups. For example, in an application that can edit multiple documents, one group holding global actions (e.g. quit, about, new), and one group per document holding actions that act on that document (eg. save, cut/copy/paste, etc). Each window's menus would be constructed from a combination of two action groups.

Accelerators

Accelerators are handled by the GTK+ accelerator map. All actions are assigned an accelerator path (which normally has the form `<Actions>/group-name/action-name`) and a shortcut is associated with this accelerator path. All menuitems and toolitems take on this accelerator path. The GTK+ accelerator map code makes sure that the correct shortcut is displayed next to the menu item.

GtkActionGroup as GtkBuildable

The GtkActionGroup implementation of the GtkBuildable interface accepts GtkAction objects as <child> elements in UI definitions.

Note that it is probably more common to define actions and action groups in the code, since they are directly related to what the code can do.

The GtkActionGroup implementation of the GtkBuildable interface supports a custom <accelerator> element, which has attributes named "key" and "modifiers" and allows to specify accelerators. This is similar to the <accelerator> element of GtkWidget, the main difference is that it doesn't allow you to specify a signal.

A GtkDialog UI definition fragment.

```
<object class="GtkActionGroup"
id="actiongroup">
  <child>
      <object class="GtkAction" id="About">
          <property
name="name">About</property>
          <property name="stock_id">gtk-
about</property>
          <signal handler="about_activate"
name="activate"/>
      </object>
      <accelerator key="F1"
modifiers="GDK_CONTROL_MASK |
GDK_SHIFT_MASK"/>
  </child>
</object>
```

Functions

gtk_action_group_new ()

```
GtkActionGroup *
gtk_action_group_new (const gchar *name);
```

`gtk_action_group_new` has been deprecated since version 3.10 and should not be used in newly-written code.

Creates a new GtkActionGroup object. The name of the action group is used when associating keybindings with the actions.

Parameters

name	the name of the action group.

Returns

the new GtkActionGroup

Since: 2.4

---

gtk_action_group_get_name ()

```
const gchar *
gtk_action_group_get_name (GtkActionGroup *action_group);
```

`gtk_action_group_get_name` has been deprecated since version 3.10 and should not be used in newly-written code.

Gets the name of the action group.

Parameters

action_group	the action group

Returns

the name of the action group.

Since: 2.4

---

gtk_action_group_get_sensitive ()

```
gboolean
gtk_action_group_get_sensitive (GtkActionGroup *action_group);
```

`gtk_action_group_get_sensitive` has been deprecated since version 3.10 and should not be used in newly-written code.

Returns `TRUE` if the group is sensitive. The constituent actions can only be logically sensitive (see `gtk_action_is_sensitive()`) if they are sensitive (see `gtk_action_get_sensitive()`) and their group is sensitive.

Parameters

action_group	the action group

Returns

`TRUE` if the group is sensitive.

Since: 2.4

---

gtk_action_group_set_sensitive ()

```
void
gtk_action_group_set_sensitive (GtkActionGroup *action_group,
                                gboolean sensitive);
```

`gtk_action_group_set_sensitive` has been deprecated since version 3.10 and should not be used in newly-written code.

Changes the sensitivity of `action_group`

Parameters

action_group	the action group
sensitive	new sensitivity

Since: 2.4

---

gtk_action_group_get_visible ()

```
gboolean
gtk_action_group_get_visible (GtkActionGroup *action_group);
```

`gtk_action_group_get_visible` has been deprecated since version 3.10 and should not be used in newly-written code.

Returns `TRUE` if the group is visible. The constituent actions can only be logically visible (see `gtk_action_is_visible()`) if they are visible (see `gtk_action_get_visible()`) and their group is visible.

Parameters

action_group	the action group

Returns

`TRUE` if the group is visible.

Since: 2.4

---

gtk_action_group_set_visible ()

```
void
gtk_action_group_set_visible (GtkActionGroup *action_group,
                              gboolean visible);
```

`gtk_action_group_set_visible` has been deprecated since version 3.10 and should not be used in newly-written code.

Changes the visible of `action_group` .

Parameters

action_group	the action group
visible	new visiblity

Since: 2.4

---

gtk_action_group_get_accel_group ()

```
GtkAccelGroup *
gtk_action_group_get_accel_group (GtkActionGroup *action_group);
```

`gtk_action_group_get_accel_group` has been deprecated since version 3.10 and should not be used in newly-written code.

Gets the accelerator group.

Parameters

action_group	a GtkActionGroup

Returns

the accelerator group associated with this action group or `NULL` if there is none.

[transfer none]

Since: 3.6

---

gtk_action_group_set_accel_group ()

```
void
gtk_action_group_set_accel_group (GtkActionGroup *action_group,
                                  GtkAccelGroup *accel_group);
```

`gtk_action_group_set_accel_group` has been deprecated since version 3.10 and should not be used in newly-written code.

Sets the accelerator group to be used by every action in this group.

Parameters

action_group	a GtkActionGroup	
accel_group	a GtkAccelGroup to set or `NULL`.	[allow-none]

Since: 3.6

---

gtk_action_group_get_action ()

```
GtkAction *
gtk_action_group_get_action (GtkActionGroup *action_group,
                             const gchar *action_name);
```

`gtk_action_group_get_action` has been deprecated since version 3.10 and should not be used in newly-written code.

Looks up an action in the action group by name.

Parameters

action_group	the action group
action_name	the name of the action

Returns

the action, or `NULL` if no action by that name exists.

[transfer none]

Since: 2.4

---

gtk_action_group_list_actions ()

```
GList *
gtk_action_group_list_actions (GtkActionGroup *action_group);
```

`gtk_action_group_list_actions` has been deprecated since version 3.10 and should not be used in newly-written code.

Lists the actions in the action group.

Parameters

action_group	the action group

Returns

an allocated list of the action objects in the action group.

[element-type GtkAction][transfer container]

Since: 2.4

---

gtk_action_group_add_action ()

```
void
gtk_action_group_add_action (GtkActionGroup *action_group,
                             GtkAction *action);
```

`gtk_action_group_add_action` has been deprecated since version 3.10 and should not be used in newly-written code.

Adds an action object to the action group. Note that this function does not set up the accel path of the action, which can lead to problems if a user tries to modify the accelerator of a menuitem associated with the action. Therefore you must either set the accel path yourself with `gtk_action_set_accel_path()`, or use `gtk_action_group_add_action_with_accel (..., NULL)`.

Parameters

action_group	the action group
action	an action

Since: 2.4

---

gtk_action_group_add_action_with_accel ()

```
void
gtk_action_group_add_action_with_accel
```

```
                                (GtkActionGroup *action_group,
                                 GtkAction *action,
                                 const gchar *accelerator);
```

`gtk_action_group_add_action_with_accel` has been deprecated since version 3.10 and should not be used in newly-written code.

Adds an action object to the action group and sets up the accelerator.

If `accelerator` is `NULL`, attempts to use the accelerator associated with the stock_id of the action.

Accel paths are set to `<Actions>/group-name/action-name`.

Parameters

action_group	the action group	
action	the action to add	
accelerator	the accelerator for the action, in the format understood by `gtk_accelerator_parse()`, or "" for no accelerator, or `NULL` to use the stock accelerator.	[allow-none]

Since: 2.4

---

gtk_action_group_remove_action ()

```
void
gtk_action_group_remove_action (GtkActionGroup *action_group,
                                GtkAction *action);
```

`gtk_action_group_remove_action` has been deprecated since version 3.10 and should not be used in newly-written code.

Removes an action object from the action group.

Parameters

action_group	the action group
action	an action

Since: 2.4

---

gtk_action_group_add_actions ()

```
void
gtk_action_group_add_actions (GtkActionGroup *action_group,
                              const GtkActionEntry *entries,
                              guint n_entries,
                              gpointer user_data);
```

`gtk_action_group_add_actions` has been deprecated since version 3.10 and should not be used in newly-written code.

This is a convenience function to create a number of actions and add them to the action group.

The “activate” signals of the actions are connected to the callbacks and their accel paths are set to `<Actions>/group-name/action-name`.

[skip]

Parameters

action_group	the action group	
entries	an array of action descriptions.	[array length=n_entries]
n_entries	the number of entries	
user_data	data to pass to the action callbacks	

Since: 2.4

---

gtk_action_group_add_actions_full ()

```
void
gtk_action_group_add_actions_full (GtkActionGroup *action_group,
                                   const GtkActionEntry *entries,
                                   guint n_entries,
                                   gpointer user_data,
                                   GDestroyNotify destroy);
```

`gtk_action_group_add_actions_full` has been deprecated since version 3.10 and should not be used in newly-written code.

This variant of `gtk_action_group_add_actions()` adds a GDestroyNotify callback for `user_data` .

[skip]

Parameters

action_group	the action group	
entries	an array of action descriptions.	[array length=n_entries]
n_entries	the number of entries	
user_data	data to pass to the action callbacks	
destroy	destroy notification callback for `user_data` .	[nullable]

Since: 2.4

---

gtk_action_group_add_toggle_actions ()

```
void
gtk_action_group_add_toggle_actions (GtkActionGroup *action_group,
                                     const GtkToggleActionEntry *entries,
                                     guint n_entries,
                                     gpointer user_data);
```

`gtk_action_group_add_toggle_actions` has been deprecated since version 3.10 and should not be used in newly-written code.

This is a convenience function to create a number of toggle actions and add them to the action group.

The “activate” signals of the actions are connected to the callbacks and their accel paths are set to `<Actions>/group-name/action-name`.

[skip]

Parameters

action_group	the action group	
entries	an array of toggle action descriptions.	[array length=n_entries]
n_entries	the number of entries	
user_data	data to pass to the action callbacks	

Since: 2.4

---

gtk_action_group_add_toggle_actions_full ()

```
void
gtk_action_group_add_toggle_actions_full
                                (GtkActionGroup *action_group,
                                 const GtkToggleActionEntry *entries,
                                 guint n_entries,
                                 gpointer user_data,
                                 GDestroyNotify destroy);
```

`gtk_action_group_add_toggle_actions_full` has been deprecated since version 3.10 and should not be used in newly-written code.

This variant of `gtk_action_group_add_toggle_actions()` adds a GDestroyNotify callback for `user_data` .

[skip]

Parameters

action_group	the action group	
entries	an array of toggle action descriptions.	[array length=n_entries]
n_entries	the number of entries	
user_data	data to pass to the action callbacks	
destroy	destroy notification callback for `user_data` .	[nullable]

Since: 2.4

---

gtk_action_group_add_radio_actions ()

```
void
gtk_action_group_add_radio_actions (GtkActionGroup *action_group,
                                    const GtkRadioActionEntry *entries,
                                    guint n_entries,
                                    gint value,
                                    GCallback on_change,
```

```
                                gpointer user_data);
```

`gtk_action_group_add_radio_actions` has been deprecated since version 3.10 and should not be used in newly-written code.

This is a convenience routine to create a group of radio actions and add them to the action group.

The “changed” signal of the first radio action is connected to the `on_change` callback and the accel paths of the actions are set to `<Actions>/group-name/action-name`.

[skip]

Parameters

action_group	the action group	
entries	an array of radio action descriptions.	[array length=n_entries]
n_entries	the number of entries	
value	the value of the action to activate initially, or -1 if no action should be activated	
on_change	the callback to connect to the changed signal	
user_data	data to pass to the action callbacks	

Since: 2.4

---

gtk_action_group_add_radio_actions_full ()

```
void
gtk_action_group_add_radio_actions_full
                               (GtkActionGroup *action_group,
                                const GtkRadioActionEntry *entries,
                                guint n_entries,
                                gint value,
                                GCallback on_change,
                                gpointer user_data,
                                GDestroyNotify destroy);
```

`gtk_action_group_add_radio_actions_full` has been deprecated since version 3.10 and should not be used in newly-written code.

This variant of `gtk_action_group_add_radio_actions()` adds a GDestroyNotify callback for `user_data` .

[skip]

Parameters

action_group	the action group	
entries	an array of radio action descriptions.	[array length=n_entries]
n_entries	the number of entries	
value	the value of the action to activate initially, or -1 if no action should be	

	activated
on_change	the callback to connect to the changed signal
user_data	data to pass to the action callbacks
destroy	destroy notification callback for `user_data`

Since: 2.4

GtkTranslateFunc ()

```
gchar *
(*GtkTranslateFunc) (const gchar *path,
                     gpointer func_data);
```

`GtkTranslateFunc` has been deprecated since version 3.10 and should not be used in newly-written code.

The function used to translate messages in e.g. GtkIconFactory and GtkActionGroup.

Parameters

path	The id of the message. In GtkActionGroup this will be a label or tooltip from a GtkActionEntry.
func_data	user data passed in when registering the function. [closure]

Returns

the translated message

---

gtk_action_group_set_translate_func ()

```
void
gtk_action_group_set_translate_func (GtkActionGroup *action_group,
                                     GtkTranslateFunc func,
                                     gpointer data,
                                     GDestroyNotify notify);
```

`gtk_action_group_set_translate_func` has been deprecated since version 3.10 and should not be used in newly-written code.

Sets a function to be used for translating the `label` and `tooltip` of GtkActionEntrys added by `gtk_action_group_add_actions()`.

If you're using `gettext()`, it is enough to set the translation domain with `gtk_action_group_set_translation_domain()`.

Parameters

action_group	a GtkActionGroup

func	a GtkTranslateFunc
data	data to be passed to `func` and `notify`
notify	a GDestroyNotify function to be called when `action_group` is destroyed and when the translation function is changed again

Since: 2.4

---

gtk_action_group_set_translation_domain ()

```
void
gtk_action_group_set_translation_domain
                                (GtkActionGroup *action_group,
                                 const gchar *domain);
```

`gtk_action_group_set_translation_domain` has been deprecated since version 3.10 and should not be used in newly-written code.

Sets the translation domain and uses `g_dgettext()` for translating the `label` and `tooltip` of GtkActionEntrys added by `gtk_action_group_add_actions()`.

If you're not using `gettext()` for localization, see `gtk_action_group_set_translate_func()`.

Parameters

action_group	a GtkActionGroup	
domain	the translation domain to use for `g_dgettext()` calls, or `NULL` to use the domain set with `textdomain()`.	[allow-none]

Since: 2.4

---

gtk_action_group_translate_string ()

```
const gchar *
gtk_action_group_translate_string (GtkActionGroup *action_group,
                                   const gchar *string);
```

`gtk_action_group_translate_string` has been deprecated since version 3.10 and should not be used in newly-written code.

Translates a string using the function set with `gtk_action_group_set_translate_func()`. This is mainly intended for language bindings.

Parameters

action_group	a GtkActionGroup
string	a string

Returns

the translation of `string`

Since: 2.6

Types and Values

struct GtkActionGroup

```
struct GtkActionGroup;
```

---

struct GtkActionGroupClass

```
struct GtkActionGroupClass {
  GObjectClass parent_class;

  GtkAction *(* get_action) (GtkActionGroup *action_group,
                             const gchar    *action_name);
};
```

Members

`get_action` ()	Looks up an action in the action group by name.

---

struct GtkActionEntry

```
struct GtkActionEntry {
  const gchar     *name;
  const gchar     *stock_id;
  const gchar     *label;
  const gchar     *accelerator;
  const gchar     *tooltip;
  GCallback  callback;
};
```

`GtkActionEntry` has been deprecated since version 3.10 and should not be used in newly-written code.

GtkActionEntry structs are used with `gtk_action_group_add_actions()` to construct actions.

Members

const gchar *`name`;	The name of the action.
const gchar *`stock_id`;	The stock id for the action, or the name of an icon from the icon theme.
const gchar *`label`;	The label for the action. This field should typically be marked for

	translation, see gtk_action_group_set_translation_domain(). If label is NULL, the label of the stock item with id stock_id is used.
const gchar *accelerator;	The accelerator for the action, in the format understood by gtk_accelerator_parse().
const gchar *tooltip;	The tooltip for the action. This field should typically be marked for translation, see gtk_action_group_set_translation_domain().
GCallback callback;	The function to call when the action is activated.

---

struct GtkToggleActionEntry

```
struct GtkToggleActionEntry {
  const gchar     *name;
  const gchar     *stock_id;
  const gchar     *label;
  const gchar     *accelerator;
  const gchar     *tooltip;
  GCallback  callback;
  gboolean   is_active;
};
```

GtkToggleActionEntry has been deprecated since version 3.10 and should not be used in newly-written code.

GtkToggleActionEntry structs are used with gtk_action_group_add_toggle_actions() to construct toggle actions.

Members

const gchar *name;	The name of the action.
const gchar *stock_id;	The stock id for the action, or the name of an icon from the icon theme.
const gchar *label;	The label for the action. This field should typically be marked for translation, see gtk_action_group_set_translation_domain().
const gchar *accelerator;	The accelerator for the action, in the format understood by gtk_accelerator_parse().
const gchar *tooltip;	The tooltip for the action. This field should typically be marked for translation, see gtk_action_group_set_translation_domain().

GCallback `callback`;	The function to call when the action is activated.
gboolean `is_active`;	The initial state of the toggle action.

---

struct GtkRadioActionEntry

```
struct GtkRadioActionEntry {
  const gchar *name;
  const gchar *stock_id;
  const gchar *label;
  const gchar *accelerator;
  const gchar *tooltip;
  gint   value;
};
```

`GtkRadioActionEntry` has been deprecated since version 3.10 and should not be used in newly-written code.

GtkRadioActionEntry structs are used with `gtk_action_group_add_radio_actions()` to construct groups of radio actions.

Members

const gchar *`name`;	The name of the action.
const gchar *`stock_id`;	The stock id for the action, or the name of an icon from the icon theme.
const gchar *`label`;	The label for the action. This field should typically be marked for translation, see `gtk_action_group_set_translation_domain()`.
const gchar *`accelerator`;	The accelerator for the action, in the format understood by `gtk_accelerator_parse()`.
const gchar *`tooltip`;	The tooltip for the action. This field should typically be marked for translation, see `gtk_action_group_set_translation_domain()`.
gint `value`;	The value to set on the radio action. See `gtk_radio_action_get_current_value()`.

Property Details

The `"accel-group"` property

```
  "accel-group"              GtkAccelGroup *
```

The accelerator group the actions of this group should use.

GtkActionGroup:accel-group has been deprecated since version 3.10 and should not be used in newly-written code.

Flags: Read / Write

The “name” property

“name” gchar *

A name for the action.

GtkActionGroup:name has been deprecated since version 3.10 and should not be used in newly-written code.

Flags: Read / Write / Construct Only

Default value: NULL

The “sensitive” property

“sensitive” gboolean

Whether the action group is enabled.

GtkActionGroup:sensitive has been deprecated since version 3.10 and should not be used in newly-written code.

Flags: Read / Write

Default value: TRUE

The “visible” property

“visible” gboolean

Whether the action group is visible.

GtkActionGroup:visible has been deprecated since version 3.10 and should not be used in newly-written code.

Flags: Read / Write

Default value: TRUE

Signal Details

The “connect-proxy” signal

```
void
user_function (GtkActionGroup *action_group,
               GtkAction      *action,
```

```
                GtkWidget      *proxy,
                gpointer        user_data)
```

The ::connect-proxy signal is emitted after connecting a proxy to an action in the group. Note that the proxy may have been connected to a different action before.

This is intended for simple customizations for which a custom action class would be too clumsy, e.g. showing tooltips for menuitems in the statusbar.

GtkUIManager proxies the signal and provides global notification just before any action is connected to a proxy, which is probably more convenient to use.

`GtkActionGroup::connect-proxy` has been deprecated since version 3.10 and should not be used in newly-written code.

Parameters

action_group	the group
action	the action
proxy	the proxy
user_data	user data set when the signal handler was connected.

Since: 2.4

The `"disconnect-proxy"` signal

```
void
user_function (GtkActionGroup *action_group,
               GtkAction      *action,
               GtkWidget      *proxy,
               gpointer        user_data)
```

The ::disconnect-proxy signal is emitted after disconnecting a proxy from an action in the group.

GtkUIManager proxies the signal and provides global notification just before any action is connected to a proxy, which is probably more convenient to use.

`GtkActionGroup::disconnect-proxy` has been deprecated since version 3.10 and should not be used in newly-written code.

Parameters

action_group	the group
action	the action
proxy	the proxy
user_data	user data set when the signal handler was connected.

Since: 2.4

The “post-activate” signal

```
void
user_function (GtkActionGroup *action_group,
               GtkAction      *action,
               gpointer        user_data)
```

The ::post-activate signal is emitted just after the `action` in the `action_group` is activated

This is intended for GtkUIManager to proxy the signal and provide global notification just after any action is activated.

`GtkActionGroup::post-activate` has been deprecated since version 3.10 and should not be used in newly-written code.

Parameters

action_group	the group
action	the action
user_data	user data set when the signal handler was connected.

Since: 2.4

The “pre-activate” signal

```
void
user_function (GtkActionGroup *action_group,
               GtkAction      *action,
               gpointer        user_data)
```

The ::pre-activate signal is emitted just before the `action` in the `action_group` is activated

This is intended for GtkUIManager to proxy the signal and provide global notification just before any action is activated.

`GtkActionGroup::pre-activate` has been deprecated since version 3.10 and should not be used in newly-written code.

Parameters

action_group	the group
action	the action
user_data	user data set when the signal handler was connected.

Since: 2.4

GtkAction

GtkAction — A deprecated action which can be triggered by a menu or toolbar item

Functions

GtkAction *	gtk_action_new ()
const gchar *	gtk_action_get_name ()
gboolean	gtk_action_is_sensitive ()
gboolean	gtk_action_get_sensitive ()
void	gtk_action_set_sensitive ()
gboolean	gtk_action_is_visible ()
gboolean	gtk_action_get_visible ()
void	gtk_action_set_visible ()
void	gtk_action_activate ()
GtkWidget *	gtk_action_create_icon ()
GtkWidget *	gtk_action_create_menu_item ()
GtkWidget *	gtk_action_create_tool_item ()
GtkWidget *	gtk_action_create_menu ()
GSList *	gtk_action_get_proxies ()
void	gtk_action_connect_accelerator ()
void	gtk_action_disconnect_accelerator ()
void	gtk_action_block_activate ()
void	gtk_action_unblock_activate ()
gboolean	gtk_action_get_always_show_image ()
void	gtk_action_set_always_show_image ()
const gchar *	gtk_action_get_accel_path ()
void	gtk_action_set_accel_path ()
GClosure *	gtk_action_get_accel_closure ()
void	gtk_action_set_accel_group ()
void	gtk_action_set_label ()
const gchar *	gtk_action_get_label ()
void	gtk_action_set_short_label ()
const gchar *	gtk_action_get_short_label ()
void	gtk_action_set_tooltip ()
const gchar *	gtk_action_get_tooltip ()
void	gtk_action_set_stock_id ()
const gchar *	gtk_action_get_stock_id ()
void	gtk_action_set_gicon ()
GIcon *	gtk_action_get_gicon ()
void	gtk_action_set_icon_name ()
const gchar *	gtk_action_get_icon_name ()
void	gtk_action_set_visible_horizontal ()
gboolean	gtk_action_get_visible_horizontal ()
void	gtk_action_set_visible_vertical ()
gboolean	gtk_action_get_visible_vertical ()
void	gtk_action_set_is_important ()
gboolean	gtk_action_get_is_important ()

Properties

GtkActionGroup *	action-group	Read / Write
gboolean	always-show-image	Read / Write / Construct
GIcon *	gicon	Read / Write
gboolean	hide-if-empty	Read / Write
gchar *	icon-name	Read / Write

gboolean	is-important	Read / Write
gchar *	label	Read / Write
gchar *	name	Read / Write / Construct Only
gboolean	sensitive	Read / Write
gchar *	short-label	Read / Write
gchar *	stock-id	Read / Write
gchar *	tooltip	Read / Write
gboolean	visible	Read / Write
gboolean	visible-horizontal	Read / Write
gboolean	visible-overflown	Read / Write
gboolean	visible-vertical	Read / Write

Signals

void	activate	No Recursion

Types and Values

struct	GtkAction
struct	GtkActionClass

Object Hierarchy

```
GObject
└── GtkAction
    ├── GtkToggleAction
    └── GtkRecentAction
```

Implemented Interfaces

GtkAction implements GtkBuildable.

Includes

```
#include <gtk/gtk.h>
```

Description

> In GTK+ 3.10, GtkAction has been deprecated. Use GAction instead, and associate actions with GtkActionable widgets. Use GMenuModel for creating menus with `gtk_menu_new_from_model()`.

Actions represent operations that the user can be perform, along with some information how it should be presented in the interface. Each action provides methods to create icons, menu items and toolbar items representing itself.

As well as the callback that is called when the action gets activated, the following also gets associated with the action:

- a name (not translated, for path lookup)

- a label (translated, for display)
- an accelerator
- whether label indicates a stock id
- a tooltip (optional, translated)
- a toolbar label (optional, shorter than label)

The action will also have some state information:

- visible (shown/hidden)
- sensitive (enabled/disabled)

Apart from regular actions, there are toggle actions, which can be toggled between two states and radio actions, of which only one in a group can be in the “active” state. Other actions can be implemented as GtkAction subclasses.

Each action can have one or more proxy widgets. To act as an action proxy, widget needs to implement GtkActivatable interface. Proxies mirror the state of the action and should change when the action’s state changes. Properties that are always mirrored by proxies are “sensitive” and “visible”. “gicon”, “icon-name”, “label”, “short-label” and “stock-id” properties are only mirorred if proxy widget has “use-action-appearance” property set to `TRUE`.

When the proxy is activated, it should activate its action.

Functions

gtk_action_new ()

```
GtkAction *
gtk_action_new (const gchar *name,
                const gchar *label,
                const gchar *tooltip,
                const gchar *stock_id);
```

`gtk_action_new` has been deprecated since version 3.10 and should not be used in newly-written code.

Use GAction instead, associating it to a widget with GtkActionable or creating a GtkMenu with `gtk_menu_new_from_model()`

Creates a new GtkAction object. To add the action to a GtkActionGroup and set the accelerator for the action, call `gtk_action_group_add_action_with_accel()`. See the UI Definition section for information on allowed action names.

Parameters

name	A unique name for the action	
label	the label displayed in menu items and on buttons, or `NULL`.	[allow-none]
tooltip	a tooltip for the action, or `NULL`.	[allow-none]
stock_id	the stock icon to display in widgets representing the action, or `NULL`.	[allow-none]

Returns

a new GtkAction

Since: 2.4

---

gtk_action_get_name ()

```
const gchar *
gtk_action_get_name (GtkAction *action);
```

`gtk_action_get_name` has been deprecated since version 3.10 and should not be used in newly-written code.

Use `g_action_get_name()` on a GAction instead

Returns the name of the action.

Parameters

action	the action object

Returns

the name of the action. The string belongs to GTK+ and should not be freed.

Since: 2.4

---

gtk_action_is_sensitive ()

```
gboolean
gtk_action_is_sensitive (GtkAction *action);
```

`gtk_action_is_sensitive` has been deprecated since version 3.10 and should not be used in newly-written code.

Use `g_action_get_enabled()` on a GAction instead

Returns whether the action is effectively sensitive.

Parameters

action	the action object

Returns

`TRUE` if the action and its associated action group are both sensitive.

Since: 2.4

---

gtk_action_get_sensitive ()

```
gboolean
gtk_action_get_sensitive (GtkAction *action);
```

`gtk_action_get_sensitive` has been deprecated since version 3.10 and should not be used in newly-written code.

Use `g_action_get_enabled()` on a GAction instead

Returns whether the action itself is sensitive. Note that this doesn't necessarily mean effective sensitivity. See `gtk_action_is_sensitive()` for that.

Parameters

action	the action object

Returns

`TRUE` if the action itself is sensitive.

Since: 2.4

---

gtk_action_set_sensitive ()

```
void
gtk_action_set_sensitive (GtkAction *action,
                          gboolean sensitive);
```

`gtk_action_set_sensitive` has been deprecated since version 3.10 and should not be used in newly-written code.

Use `g_simple_action_set_enabled()` on a GSimpleAction instead

Sets the :sensitive property of the action to `sensitive` . Note that this doesn't necessarily mean effective sensitivity. See `gtk_action_is_sensitive()` for that.

Parameters

action	the action object
sensitive	`TRUE` to make the action sensitive

Since: 2.6

---

gtk_action_is_visible ()

```
gboolean
gtk_action_is_visible (GtkAction *action);
```

`gtk_action_is_visible` has been deprecated since version 3.10 and should not be used in newly-written code.

Use GAction instead, and control and monitor the state of GtkActionable widgets directly

Returns whether the action is effectively visible.

Parameters

action	the action object

Returns

`TRUE` if the action and its associated action group are both visible.

Since: 2.4

---

gtk_action_get_visible ()

```
gboolean
gtk_action_get_visible (GtkAction *action);
```

`gtk_action_get_visible` has been deprecated since version 3.10 and should not be used in newly-written code.

Use GAction instead, and control and monitor the state of GtkActionable widgets directly

Returns whether the action itself is visible. Note that this doesn't necessarily mean effective visibility. See `gtk_action_is_sensitive()` for that.

Parameters

action	the action object

Returns

`TRUE` if the action itself is visible.

Since: 2.4

---

gtk_action_set_visible ()

```
void
gtk_action_set_visible (GtkAction *action,
                        gboolean visible);
```

`gtk_action_set_visible` has been deprecated since version 3.10 and should not be used in newly-written code.

Use GAction instead, and control and monitor the state of GtkActionable widgets directly

Sets the :visible property of the action to `visible` . Note that this doesn't necessarily mean effective visibility. See `gtk_action_is_visible()` for that.

Parameters

action	the action object
visible	`TRUE` to make the action visible

Since: 2.6

---

gtk_action_activate ()

```
void
gtk_action_activate (GtkAction *action);
```

`gtk_action_activate` has been deprecated since version 3.10 and should not be used in newly-written code.

Use `g_action_group_activate_action()` on a GAction instead

Emits the “activate” signal on the specified action, if it isn't insensitive. This gets called by the proxy widgets when they get activated.

It can also be used to manually activate an action.

Parameters

action	the action object

Since: 2.4

---

gtk_action_create_icon ()

```
GtkWidget *
gtk_action_create_icon (GtkAction *action,
                        GtkIconSize icon_size);
```

`gtk_action_create_icon` has been deprecated since version 3.10 and should not be used in newly-written code.

Use `g_menu_item_set_icon()` to set an icon on a GMenuItem, or `gtk_container_add()` to add a GtkImage to a GtkButton

This function is intended for use by action implementations to create icons displayed in the proxy widgets.

Parameters

action	the action object	
icon_size	the size of the icon (GtkIconSize) that should be created.	[type int]

Returns

a widget that displays the icon for this action.

[transfer none]

Since: 2.4

gtk_action_create_menu_item ()

```
GtkWidget *
gtk_action_create_menu_item (GtkAction *action);
```

`gtk_action_create_menu_item` has been deprecated since version 3.10 and should not be used in newly-written code.

Use `g_menu_item_new()` and associate it with a GAction instead.

Creates a menu item widget that proxies for the given action.

Parameters

action	the action object

Returns

a menu item connected to the action.

[transfer none]

Since: 2.4

---

gtk_action_create_tool_item ()

```
GtkWidget *
gtk_action_create_tool_item (GtkAction *action);
```

`gtk_action_create_tool_item` has been deprecated since version 3.10 and should not be used in newly-written code.

Use a GtkToolItem and associate it with a GAction using `gtk_actionable_set_action_name()` instead

Creates a toolbar item widget that proxies for the given action.

Parameters

action	the action object

Returns

a toolbar item connected to the action.

[transfer none]

Since: 2.4

---

gtk_action_create_menu ()

```
GtkWidget *
gtk_action_create_menu (GtkAction *action);
```

`gtk_action_create_menu` has been deprecated since version 3.10 and should not be used in newly-written code.

Use GAction and GMenuModel instead, and create a GtkMenu with `gtk_menu_new_from_model()`

If `action` provides a GtkMenu widget as a submenu for the menu item or the toolbar item it creates, this function returns an instance of that menu.

Parameters

action	a GtkAction

Returns

the menu item provided by the action, or `NULL`.

[transfer none]

Since: 2.12

---

gtk_action_get_proxies ()

```
GSList *
gtk_action_get_proxies (GtkAction *action);
```

`gtk_action_get_proxies` has been deprecated since version 3.10 and should not be used in newly-written code.

Returns the proxy widgets for an action. See also `gtk_activatable_get_related_action()`.

Parameters

action	the action object

Returns

a GSList of proxy widgets. The list is owned by GTK+ and must not be modified.

[element-type GtkWidget][transfer none]

Since: 2.4

---

gtk_action_connect_accelerator ()

```
void
gtk_action_connect_accelerator (GtkAction *action);
```

`gtk_action_connect_accelerator` has been deprecated since version 3.10 and should not be used in newly-written code.

Use GAction and the accelerator group on an associated GtkMenu instead

Installs the accelerator for `action` if `action` has an accel path and group. See `gtk_action_set_accel_path()` and `gtk_action_set_accel_group()`

Since multiple proxies may independently trigger the installation of the accelerator, the `action` counts the number of times this function has been called and doesn't remove the accelerator until `gtk_action_disconnect_accelerator()` has been called as many times.

Parameters

action	a GtkAction

Since: 2.4

---

gtk_action_disconnect_accelerator ()

```
void
gtk_action_disconnect_accelerator (GtkAction *action);
```

`gtk_action_disconnect_accelerator` has been deprecated since version 3.10 and should not be used in newly-written code.

Use GAction and the accelerator group on an associated GtkMenu instead

Undoes the effect of one call to `gtk_action_connect_accelerator()`.

Parameters

action	a GtkAction

Since: 2.4

---

gtk_action_block_activate ()

```
void
gtk_action_block_activate (GtkAction *action);
```

`gtk_action_block_activate` has been deprecated since version 3.10 and should not be used in newly-written code.

Use `g_simple_action_set_enabled()` to disable the GSimpleAction instead

Disable activation signals from the action

This is needed when updating the state of your proxy GtkActivatable widget could result in calling `gtk_action_activate()`, this is a convenience function to avoid recursing in those cases (updating toggle state for instance).

Parameters

action	a GtkAction

Since: 2.16

---

gtk_action_unblock_activate ()

```
void
gtk_action_unblock_activate (GtkAction *action);
```

`gtk_action_unblock_activate` has been deprecated since version 3.10 and should not be used in newly-written code.

Use `g_simple_action_set_enabled()` to enable the GSimpleAction instead

Reenable activation signals from the action

Parameters

action	a GtkAction

Since: 2.16

---

gtk_action_get_always_show_image ()

```
gboolean
gtk_action_get_always_show_image (GtkAction *action);
```

`gtk_action_get_always_show_image` has been deprecated since version 3.10 and should not be used in newly-written code.

Use `g_menu_item_get_attribute_value()` on a GMenuItem instead

Returns whether `action` 's menu item proxies will always show their image, if available.

Parameters

action	a GtkAction

Returns

`TRUE` if the menu item proxies will always show their image

Since: 2.20

---

gtk_action_set_always_show_image ()

```
void
gtk_action_set_always_show_image (GtkAction *action,
                                  gboolean always_show);
```

`gtk_action_set_always_show_image` has been deprecated since version 3.10 and should not be used in newly-written code.

Use `g_menu_item_set_icon()` on a GMenuItem instead, if the item should have an image

Sets whether `action` 's menu item proxies will ignore the "gtk-menu-images" setting and always show their image, if available.

Use this if the menu item would be useless or hard to use without their image.

Parameters

action	a GtkAction
always_show	`TRUE` if menuitem proxies should always show their image

Since: 2.20

---

gtk_action_get_accel_path ()

```
const gchar *
gtk_action_get_accel_path (GtkAction *action);
```

`gtk_action_get_accel_path` has been deprecated since version 3.10 and should not be used in newly-written code.

Use GAction and the accelerator path on an associated GtkMenu instead

Returns the accel path for this action.

Parameters

action	the action object

Returns

the accel path for this action, or `NULL` if none is set. The returned string is owned by GTK+ and must not be freed or modified.

Since: 2.6

---

gtk_action_set_accel_path ()

```
void
gtk_action_set_accel_path (GtkAction *action,
                           const gchar *accel_path);
```

`gtk_action_set_accel_path` has been deprecated since version 3.10 and should not be used in newly-written code.

Use GAction and the accelerator path on an associated GtkMenu instead

Sets the accel path for this action. All proxy widgets associated with the action will have this accel path, so that their accelerators are consistent.

Note that `accel_path` string will be stored in a GQuark. Therefore, if you pass a static string, you can save some memory by interning it first with `g_intern_static_string()`.

Parameters

action	the action object
accel_path	the accelerator path

Since: 2.4

---

gtk_action_get_accel_closure ()

```
GClosure *
gtk_action_get_accel_closure (GtkAction *action);
```

`gtk_action_get_accel_closure` has been deprecated since version 3.10 and should not be used in newly-written code.

Use GAction and GtkMenu instead, which have no equivalent for getting the accel closure

Returns the accel closure for this action.

Parameters

action	the action object

Returns

the accel closure for this action. The returned closure is owned by GTK+ and must not be unreffed or modified.

[transfer none]

Since: 2.8

---

gtk_action_set_accel_group ()

```
void
gtk_action_set_accel_group (GtkAction *action,
                            GtkAccelGroup *accel_group);
```

`gtk_action_set_accel_group` has been deprecated since version 3.10 and should not be used in newly-written code.

Use GAction and the accelerator group on an associated GtkMenu instead

Sets the GtkAccelGroup in which the accelerator for this action will be installed.

Parameters

action	the action object	
accel_group	a GtkAccelGroup or `NULL`.	[allow-none]

Since: 2.4

---

gtk_action_set_label ()

```
void
gtk_action_set_label (GtkAction *action,
                      const gchar *label);
```

`gtk_action_set_label` has been deprecated since version 3.10 and should not be used in newly-written code.

Use GAction instead, and set a label on a menu item with `g_menu_item_set_label()`. For GtkActionable widgets, use the widget-specific API to set a label

Sets the label of `action` .

Parameters

action	a GtkAction
label	the label text to set

Since: 2.16

---

gtk_action_get_label ()

```
const gchar *
gtk_action_get_label (GtkAction *action);
```

`gtk_action_get_label` has been deprecated since version 3.10 and should not be used in newly-written code.

Use GAction instead, and get a label from a menu item with `g_menu_item_get_attribute_value()`. For GtkActionable widgets, use the widget-specific API to get a label

Gets the label text of `action` .

Parameters

action	a GtkAction

Returns

the label text

Since: 2.16

---

gtk_action_set_short_label ()

```
void
gtk_action_set_short_label (GtkAction *action,
                            const gchar *short_label);
```

`gtk_action_set_short_label` has been deprecated since version 3.10 and should not be used in newly-written code.

Use GAction instead, which has no equivalent of short labels

Sets a shorter label text on `action` .

Parameters

action	a GtkAction
short_label	the label text to set

Since: 2.16

---

gtk_action_get_short_label ()

```
const gchar *
gtk_action_get_short_label (GtkAction *action);
```

`gtk_action_get_short_label` has been deprecated since version 3.10 and should not be used in newly-written code.

Use GAction instead, which has no equivalent of short labels

Gets the short label text of `action` .

Parameters

action	a GtkAction

Returns

the short label text.

Since: 2.16

---

gtk_action_set_tooltip ()

```
void
gtk_action_set_tooltip (GtkAction *action,
                        const gchar *tooltip);
```

`gtk_action_set_tooltip` has been deprecated since version 3.10 and should not be used in newly-written code.

Use GAction instead, and set tooltips on associated GtkActionable widgets with `gtk_widget_set_tooltip_text()`

Sets the tooltip text on `action`

Parameters

action	a GtkAction
tooltip	the tooltip text

Since: 2.16

---

gtk_action_get_tooltip ()

```
const gchar *
gtk_action_get_tooltip (GtkAction *action);
```

`gtk_action_get_tooltip` has been deprecated since version 3.10 and should not be used in newly-written code.

Use GAction instead, and get tooltips from associated GtkActionable widgets with `gtk_widget_get_tooltip_text()`

Gets the tooltip text of `action` .

Parameters

action	a GtkAction

Returns

the tooltip text

Since: 2.16

---

gtk_action_set_stock_id ()

```
void
gtk_action_set_stock_id (GtkAction *action,
                         const gchar *stock_id);
```

`gtk_action_set_stock_id` has been deprecated since version 3.10 and should not be used in newly-written code.

Use GAction instead, which has no equivalent of stock items

Sets the stock id on `action`

Parameters

action	a GtkAction
stock_id	the stock id

Since: 2.16

gtk_action_get_stock_id ()

```
const gchar *
gtk_action_get_stock_id (GtkAction *action);
```

`gtk_action_get_stock_id` has been deprecated since version 3.10 and should not be used in newly-written code.

Use GAction instead, which has no equivalent of stock items

Gets the stock id of `action` .

Parameters

action	a GtkAction

Returns

the stock id

Since: 2.16

---

gtk_action_set_gicon ()

```
void
gtk_action_set_gicon (GtkAction *action,
                      GIcon *icon);
```

`gtk_action_set_gicon` has been deprecated since version 3.10 and should not be used in newly-written code.

Use GAction instead, and `g_menu_item_set_icon()` to set an icon on a GMenuItem associated with a GAction, or `gtk_container_add()` to add a GtkImage to a GtkButton

Sets the icon of `action` .

Parameters

action	a GtkAction
icon	the GIcon to set

Since: 2.16

---

gtk_action_get_gicon ()

```
GIcon *
gtk_action_get_gicon (GtkAction *action);
```

`gtk_action_get_gicon` has been deprecated since version 3.10 and should not be used in newly-written code.

Use GAction instead, and `g_menu_item_get_attribute_value()` to get an icon from a GMenuItem

associated with a GAction

Gets the gicon of `action` .

Parameters

action	a GtkAction

Returns

The action’s GIcon if one is set.

[transfer none]

Since: 2.16

---

gtk_action_set_icon_name ()

```
void
gtk_action_set_icon_name (GtkAction *action,
                          const gchar *icon_name);
```

`gtk_action_set_icon_name` has been deprecated since version 3.10 and should not be used in newly-written code.

Use GAction instead, and `g_menu_item_set_icon()` to set an icon on a GMenuItem associated with a GAction, or `gtk_container_add()` to add a GtkImage to a GtkButton

Sets the icon name on `action`

Parameters

action	a GtkAction
icon_name	the icon name to set

Since: 2.16

---

gtk_action_get_icon_name ()

```
const gchar *
gtk_action_get_icon_name (GtkAction *action);
```

`gtk_action_get_icon_name` has been deprecated since version 3.10 and should not be used in newly-written code.

Use GAction instead, and `g_menu_item_get_attribute_value()` to get an icon from a GMenuItem associated with a GAction

Gets the icon name of `action` .

Parameters

action	a GtkAction

Returns

the icon name

Since: 2.16

---

gtk_action_set_visible_horizontal ()

```
void
gtk_action_set_visible_horizontal (GtkAction *action,
                                   gboolean visible_horizontal);
```

`gtk_action_set_visible_horizontal` has been deprecated since version 3.10 and should not be used in newly-written code.

Use GAction instead, and control and monitor the visibility of associated widgets and menu items directly

Sets whether `action` is visible when horizontal

Parameters

action	a GtkAction
visible_horizontal	whether the action is visible horizontally

Since: 2.16

---

gtk_action_get_visible_horizontal ()

```
gboolean
gtk_action_get_visible_horizontal (GtkAction *action);
```

`gtk_action_get_visible_horizontal` has been deprecated since version 3.10 and should not be used in newly-written code.

Use GAction instead, and control and monitor the visibility of associated widgets and menu items directly

Checks whether `action` is visible when horizontal

Parameters

action	a GtkAction

Returns

whether `action` is visible when horizontal

Since: 2.16

gtk_action_set_visible_vertical ()

```
void
gtk_action_set_visible_vertical (GtkAction *action,
                                 gboolean visible_vertical);
```

`gtk_action_set_visible_vertical` has been deprecated since version 3.10 and should not be used in newly-written code.

Use GAction instead, and control and monitor the visibility of associated widgets and menu items directly

Sets whether `action` is visible when vertical

Parameters

action	a GtkAction
visible_vertical	whether the action is visible vertically

Since: 2.16

---

gtk_action_get_visible_vertical ()

```
gboolean
gtk_action_get_visible_vertical (GtkAction *action);
```

`gtk_action_get_visible_vertical` has been deprecated since version 3.10 and should not be used in newly-written code.

Use GAction instead, and control and monitor the visibility of associated widgets and menu items directly

Checks whether `action` is visible when horizontal

Parameters

action	a GtkAction

Returns

whether `action` is visible when horizontal

Since: 2.16

---

gtk_action_set_is_important ()

```
void
gtk_action_set_is_important (GtkAction *action,
                             gboolean is_important);
```

`gtk_action_set_is_important` has been deprecated since version 3.10 and should not be used in newly-written code.

Use GAction instead, and control and monitor whether labels are shown directly

Sets whether the action is important, this attribute is used primarily by toolbar items to decide whether to show a label or not.

Parameters

action	the action object
is_important	`TRUE` to make the action important

Since: 2.16

---

gtk_action_get_is_important ()

```
gboolean
gtk_action_get_is_important (GtkAction *action);
```

`gtk_action_get_is_important` has been deprecated since version 3.10 and should not be used in newly-written code.

Use GAction instead, and control and monitor whether labels are shown directly

Checks whether `action` is important or not

Parameters

action	a GtkAction

Returns

whether `action` is important

Since: 2.16

Types and Values

struct GtkAction

```
struct GtkAction;
```

---

struct GtkActionClass

```
struct GtkActionClass {
  GObjectClass parent_class;
```

```
  /* activation signal */
  void       (* activate)           (GtkAction    *action);
};
```

Members

`activate` ()	Signal emitted when the action is activated.

Property Details

The “action-group” property

```
  “action-group”             GtkActionGroup *
```

The GtkActionGroup this GtkAction is associated with, or NULL (for internal use).

`GtkAction:action-group` has been deprecated since version 3.10 and should not be used in newly-written code.

Lookup the GAction using `g_action_map_lookup_action()` instead

Flags: Read / Write

The “always-show-image” property

```
  “always-show-image”        gboolean
```

If `TRUE`, the action's menu item proxies will ignore the “gtk-menu-images” setting and always show their image, if available.

Use this property if the menu item would be useless or hard to use without their image.

`GtkAction:always-show-image` has been deprecated since version 3.10 and should not be used in newly-written code.

There is no corresponding replacement when using GAction

Flags: Read / Write / Construct

Default value: FALSE

Since: 2.20

The “gicon” property

```
  “gicon”                    GIcon *
```

The GIcon displayed in the GtkAction.

Note that the stock icon is preferred, if the “stock-id” property holds the id of an existing stock icon.

This is an appearance property and thus only applies if “use-action-appearance” is `TRUE`.

`GtkAction:gicon` has been deprecated since version 3.10 and should not be used in newly-written code.

Use the "icon" attribute on a GMenuItem instead

Flags: Read / Write

Since: 2.16

The “hide-if-empty” property

```
  “hide-if-empty”            gboolean
```

When TRUE, empty menu proxies for this action are hidden.

`GtkAction:hide-if-empty` has been deprecated since version 3.10 and should not be used in newly-written code.

There is no corresponding replacement when using GAction

Flags: Read / Write

Default value: TRUE

The “icon-name” property

```
  “icon-name”                gchar *
```

The name of the icon from the icon theme.

Note that the stock icon is preferred, if the “stock-id” property holds the id of an existing stock icon, and the GIcon is preferred if the “gicon” property is set.

This is an appearance property and thus only applies if “use-action-appearance” is `TRUE`.

`GtkAction:icon-name` has been deprecated since version 3.10 and should not be used in newly-written code.

Use the "icon" attribute on a GMenuItem instead

Flags: Read / Write

Default value: NULL

Since: 2.10

The “is-important” property

```
  “is-important”             gboolean
```

Whether the action is considered important. When TRUE, toolitem proxies for this action show text in GTK_TOOLBAR_BOTH_HORIZ mode.

`GtkAction:is-important` has been deprecated since version 3.10 and should not be used in newly-written code.

There is no corresponding replacement when using GAction

Flags: Read / Write

Default value: FALSE

The “label” property

“label” gchar *

The label used for menu items and buttons that activate this action. If the label is NULL, GTK+ uses the stock label specified via the stock-id property.

This is an appearance property and thus only applies if “use-action-appearance” is TRUE.

GtkAction:label has been deprecated since version 3.10 and should not be used in newly-written code.

Use the "label" attribute on GMenuItem instead

Flags: Read / Write

Default value: NULL

The “name” property

“name” gchar *

A unique name for the action.

GtkAction:name has been deprecated since version 3.10 and should not be used in newly-written code.

Use “name” instead

Flags: Read / Write / Construct Only

Default value: NULL

The “sensitive” property

“sensitive” gboolean

Whether the action is enabled.

GtkAction:sensitive has been deprecated since version 3.10 and should not be used in newly-written code.

Use “enabled” and “enabled” instead

Flags: Read / Write

Default value: TRUE

The “short-label” property

“short-label” gchar *

A shorter label that may be used on toolbar buttons.

This is an appearance property and thus only applies if “use-action-appearance” is TRUE.

GtkAction:short-label has been deprecated since version 3.10 and should not be used in newly-written code.

There is no corresponding replacement when using GAction

Flags: Read / Write

Default value: NULL

The “stock-id” property

“stock-id” gchar *

The stock icon displayed in widgets representing this action.

This is an appearance property and thus only applies if “use-action-appearance” is TRUE.

GtkAction:stock-id has been deprecated since version 3.10 and should not be used in newly-written code.

There is no corresponding replacement when using GAction

Flags: Read / Write

Default value: NULL

The “tooltip” property

“tooltip” gchar *

A tooltip for this action.

GtkAction:tooltip has been deprecated since version 3.10 and should not be used in newly-written code.

Use gtk_widget_set_tooltip_text() instead

Flags: Read / Write

Default value: NULL

The “visible” property

“visible” gboolean

Whether the action is visible.

GtkAction:visible has been deprecated since version 3.10 and should not be used in newly-written code.

There is no corresponding replacement when using GAction

Flags: Read / Write

Default value: TRUE

The “visible-horizontal” property

“visible-horizontal” gboolean

Whether the toolbar item is visible when the toolbar is in a horizontal orientation.

GtkAction:visible-horizontal has been deprecated since version 3.10 and should not be used in newly-written code.

There is no corresponding replacement when using GAction

Flags: Read / Write

Default value: TRUE

The “visible-overflown” property

“visible-overflown” gboolean

When TRUE, toolitem proxies for this action are represented in the toolbar overflow menu.

GtkAction:visible-overflown has been deprecated since version 3.10 and should not be used in newly-written code.

There is no corresponding replacement when using GAction

Flags: Read / Write

Default value: TRUE

Since: 2.6

The “visible-vertical” property

“visible-vertical” gboolean

Whether the toolbar item is visible when the toolbar is in a vertical orientation.

GtkAction:visible-vertical has been deprecated since version 3.10 and should not be used in newly-written code.

There is no corresponding replacement when using GAction

Flags: Read / Write

Default value: TRUE

Signal Details

The “activate” signal

void

```
user_function (GtkAction *action,
               gpointer   user_data)
```

The "activate" signal is emitted when the action is activated.

`GtkAction::activate` has been deprecated since version 3.10 and should not be used in newly-written code.

Use “activate” instead

Parameters

action	the GtkAction
user_data	user data set when the signal handler was connected.

Flags: No Recursion

Since: 2.4

See Also

GtkActionGroup, GtkUIManager, GtkActivatable

GtkToggleAction

GtkToggleAction — An action which can be toggled between two states

Functions

GtkToggleAction *	gtk_toggle_action_new ()
void	gtk_toggle_action_toggled ()
void	gtk_toggle_action_set_active ()
gboolean	gtk_toggle_action_get_active ()
void	gtk_toggle_action_set_draw_as_radio ()
gboolean	gtk_toggle_action_get_draw_as_radio ()

Properties

gboolean	active	Read / Write
gboolean	draw-as-radio	Read / Write

Signals

void	toggled	Run First

Types and Values

struct	GtkToggleAction

Object Hierarchy

```
GObject
└── GtkAction
    └── GtkToggleAction
        └── GtkRadioAction
```

Implemented Interfaces

GtkToggleAction implements GtkBuildable.

Includes

```
#include <gtk/gtk.h>
```

Description

A GtkToggleAction corresponds roughly to a GtkCheckMenuItem. It has an “active” state specifying whether the action has been checked or not.

Functions

gtk_toggle_action_new ()

```
GtkToggleAction *
gtk_toggle_action_new (const gchar *name,
                       const gchar *label,
                       const gchar *tooltip,
                       const gchar *stock_id);
```

`gtk_toggle_action_new` has been deprecated since version 3.10 and should not be used in newly-written code.

Creates a new GtkToggleAction object. To add the action to a GtkActionGroup and set the accelerator for the action, call `gtk_action_group_add_action_with_accel()`.

Parameters

name	A unique name for the action	
label	The label displayed in menu items and on buttons, or `NULL`.	[allow-none]
tooltip	A tooltip for the action, or `NULL`.	[allow-none]
stock_id	The stock icon to display in widgets representing the action, or `NULL`.	[allow-none]

Returns

a new GtkToggleAction

Since: 2.4

---

gtk_toggle_action_toggled ()

```
void
gtk_toggle_action_toggled (GtkToggleAction *action);
```

`gtk_toggle_action_toggled` has been deprecated since version 3.10 and should not be used in newly-written code.

Emits the “toggled” signal on the toggle action.

Parameters

action	the action object

Since: 2.4

---

gtk_toggle_action_set_active ()

```
void
gtk_toggle_action_set_active (GtkToggleAction *action,
                              gboolean is_active);
```

`gtk_toggle_action_set_active` has been deprecated since version 3.10 and should not be used in newly-written code.

Sets the checked state on the toggle action.

Parameters

action	the action object
is_active	whether the action should be checked or not

Since: 2.4

---

gtk_toggle_action_get_active ()

```
gboolean
gtk_toggle_action_get_active (GtkToggleAction *action);
```

`gtk_toggle_action_get_active` has been deprecated since version 3.10 and should not be used in newly-written code.

Returns the checked state of the toggle action.

Parameters

action	the action object

Returns

the checked state of the toggle action

Since: 2.4

---

gtk_toggle_action_set_draw_as_radio ()

```
void
gtk_toggle_action_set_draw_as_radio (GtkToggleAction *action,
                                     gboolean draw_as_radio);
```

`gtk_toggle_action_set_draw_as_radio` has been deprecated since version 3.10 and should not be used in newly-written code.

Sets whether the action should have proxies like a radio action.

Parameters

action	the action object
draw_as_radio	whether the action should have proxies like a radio action

Since: 2.4

---

gtk_toggle_action_get_draw_as_radio ()

```
gboolean
gtk_toggle_action_get_draw_as_radio (GtkToggleAction *action);
```

`gtk_toggle_action_get_draw_as_radio` has been deprecated since version 3.10 and should not be used in newly-written code.

Returns whether the action should have proxies like a radio action.

Parameters

action	the action object

Returns

whether the action should have proxies like a radio action.

Since: 2.4

Types and Values

struct GtkToggleAction

```
struct GtkToggleAction;
```

Property Details

The “active” property

```
  “active”                   gboolean
```

Whether the toggle action should be active.

`GtkToggleAction:active` has been deprecated since version 3.10 and should not be used in newly-written code.

Flags: Read / Write

Default value: FALSE

Since: 2.10

The “draw-as-radio” property

```
  “draw-as-radio”            gboolean
```

Whether the proxies for this action look like radio action proxies.

This is an appearance property and thus only applies if “use-action-appearance” is `TRUE`.

`GtkToggleAction:draw-as-radio` has been deprecated since version 3.10 and should not be used in newly-written code.

Flags: Read / Write

Default value: FALSE

Signal Details

The “toggled” signal

```
void
user_function (GtkToggleAction *toggleaction,
               gpointer         user_data)
```

Should be connected if you wish to perform an action whenever the GtkToggleAction state is changed.

`GtkToggleAction::toggled` has been deprecated since version 3.10 and should not be used in newly-written code.

Parameters

toggleaction	the object which received the signal.

user_data	user data set when the signal handler was connected.

Flags: Run First

GtkRadioAction

GtkRadioAction — An action of which only one in a group can be active

Functions

GtkRadioAction *	gtk_radio_action_new ()
GSList *	gtk_radio_action_get_group ()
void	gtk_radio_action_set_group ()
void	gtk_radio_action_join_group ()
gint	gtk_radio_action_get_current_value ()
void	gtk_radio_action_set_current_value ()

Properties

gint	current-value	Read / Write
GtkRadioAction *	group	Write
gint	value	Read / Write

Signals

void	changed	No Recursion

Types and Values

struct	GtkRadioAction

Object Hierarchy

```
GObject
└── GtkAction
    └── GtkToggleAction
        └── GtkRadioAction
```

Implemented Interfaces

GtkRadioAction implements GtkBuildable.

Includes

```
#include <gtk/gtk.h>
```

Description

A GtkRadioAction is similar to GtkRadioMenuItem. A number of radio actions can be linked together so that only one may be active at any one time.

Functions

gtk_radio_action_new ()

```
GtkRadioAction *
gtk_radio_action_new (const gchar *name,
                      const gchar *label,
                      const gchar *tooltip,
                      const gchar *stock_id,
                      gint value);
```

`gtk_radio_action_new` has been deprecated since version 3.10 and should not be used in newly-written code.

Creates a new GtkRadioAction object. To add the action to a GtkActionGroup and set the accelerator for the action, call `gtk_action_group_add_action_with_accel()`.

Parameters

name	A unique name for the action	
label	The label displayed in menu items and on buttons, or `NULL`.	[allow-none]
tooltip	A tooltip for this action, or `NULL`.	[allow-none]
stock_id	The stock icon to display in widgets representing this action, or `NULL`.	[allow-none]
value	The value which `gtk_radio_action_get_current_value()` should return if this action is selected.	

Returns

a new GtkRadioAction

Since: 2.4

---

gtk_radio_action_get_group ()

```
GSList *
gtk_radio_action_get_group (GtkRadioAction *action);
```

`gtk_radio_action_get_group` has been deprecated since version 3.10 and should not be used in newly-written code.

Returns the list representing the radio group for this object. Note that the returned list is only valid until the next change to the group.

A common way to set up a group of radio group is the following:

```
GSList *group = NULL;
GtkRadioAction *action;

while ( ...more actions to add... /)
  {
     action = gtk_radio_action_new (...);

     gtk_radio_action_set_group (action,
group);
     group = gtk_radio_action_get_group
(action);
  }
```

Parameters

action	the action object

Returns

the list representing the radio group for this object.

[element-type GtkRadioAction][transfer none]

Since: 2.4

---

gtk_radio_action_set_group ()

```
void
gtk_radio_action_set_group (GtkRadioAction *action,
                            GSList *group);
```

`gtk_radio_action_set_group` has been deprecated since version 3.10 and should not be used in newly-written code.

Sets the radio group for the radio action object.

Parameters

action	the action object	
group	a list representing a radio group, or `NULL`.	[element-type GtkRadioAction][allow-none]

Since: 2.4

---

gtk_radio_action_join_group ()

```
void
gtk_radio_action_join_group (GtkRadioAction *action,
                             GtkRadioAction *group_source);
```

`gtk_radio_action_join_group` has been deprecated since version 3.10 and should not be used in newly-written code.

Joins a radio action object to the group of another radio action object.

Use this in language bindings instead of the `gtk_radio_action_get_group()` and `gtk_radio_action_set_group()` methods

A common way to set up a group of radio actions is the following:

```
GtkRadioAction *action;
GtkRadioAction *last_action;

while ( ...more actions to add... /)
  {
     action = gtk_radio_action_new (...);

     gtk_radio_action_join_group (action,
last_action);
     last_action = action;
  }
```

Parameters

action	the action object
group_source	a radio action object whos group we are joining, or `NULL` to remove the radio action from its group. [allow-none]

Since: 3.0

---

gtk_radio_action_get_current_value ()

```
gint
gtk_radio_action_get_current_value (GtkRadioAction *action);
```

`gtk_radio_action_get_current_value` has been deprecated since version 3.10 and should not be used in newly-written code.

Obtains the value property of the currently active member of the group to which `action` belongs.

Parameters

action	a GtkRadioAction

Returns

The value of the currently active group member

Since: 2.4

---

gtk_radio_action_set_current_value ()

```
void
gtk_radio_action_set_current_value (GtkRadioAction *action,
                                    gint current_value);
```

`gtk_radio_action_set_current_value` has been deprecated since version 3.10 and should not be used in newly-written code.

Sets the currently active group member to the member with value property `current_value` .

Parameters

action	a GtkRadioAction
current_value	the new value

Since: 2.10

Types and Values

struct GtkRadioAction

```
struct GtkRadioAction;
```

Property Details

The “current-value” property

```
  “current-value”            gint
```

The value property of the currently active member of the group to which this action belongs.

`GtkRadioAction:current-value` has been deprecated since version 3.10 and should not be used in newly-written code.

Flags: Read / Write

Default value: 0

Since: 2.10

The “group” property

```
  “group”                    GtkRadioAction *
```

Sets a new group for a radio action.

`GtkRadioAction:group` has been deprecated since version 3.10 and should not be used in newly-written code.

Flags: Write

Since: 2.4

The “value” property

```
  “value”                    gint
```

The value is an arbitrary integer which can be used as a convenient way to determine which action in the group is currently active in an ::activate or ::changed signal handler. See `gtk_radio_action_get_current_value()` and GtkRadioActionEntry for convenient ways to get and set this property.

`GtkRadioAction:value` has been deprecated since version 3.10 and should not be used in newly-written code.

Flags: Read / Write

Default value: 0

Since: 2.4

Signal Details

The “changed” signal

```
void
user_function (GtkRadioAction *action,
               GtkRadioAction *current,
               gpointer        user_data)
```

The ::changed signal is emitted on every member of a radio group when the active member is changed. The signal gets emitted after the ::activate signals for the previous and current active members.

`GtkRadioAction::changed` has been deprecated since version 3.10 and should not be used in newly-written code.

Parameters

action	the action on which the signal is emitted
current	the member of `action` 's group which has just been activated
user_data	user data set when the signal handler was connected.

Flags: No Recursion

Since: 2.4

GtkRecentAction

GtkRecentAction — An action of which represents a list of recently used files

Functions

GtkAction *	gtk_recent_action_new ()

GtkAction *	gtk_recent_action_new_for_manager ()
gboolean	gtk_recent_action_get_show_numbers ()
void	gtk_recent_action_set_show_numbers ()

Properties

gboolean	show-numbers	Read / Write

Types and Values

struct	GtkRecentAction

Object Hierarchy

```
GObject
└── GtkAction
    └── GtkRecentAction
```

Implemented Interfaces

GtkRecentAction implements GtkBuildable and GtkRecentChooser.

Includes

```
#include <gtk/gtk.h>
```

Description

A GtkRecentAction represents a list of recently used files, which can be shown by widgets such as GtkRecentChooserDialog or GtkRecentChooserMenu.

To construct a submenu showing recently used files, use a GtkRecentAction as the action for a <menuitem>. To construct a menu toolbutton showing the recently used files in the popup menu, use a GtkRecentAction as the action for a <toolitem> element.

Functions

gtk_recent_action_new ()

```
GtkAction *
gtk_recent_action_new (const gchar *name,
                       const gchar *label,
                       const gchar *tooltip,
                       const gchar *stock_id);
```

`gtk_recent_action_new` has been deprecated since version 3.10 and should not be used in newly-written code.

Creates a new GtkRecentAction object. To add the action to a GtkActionGroup and set the accelerator for the action, call `gtk_action_group_add_action_with_accel()`.

Parameters

name	a unique name for the action	
label	the label displayed in menu items and on buttons, or `NULL`.	[allow-none]
tooltip	a tooltip for the action, or `NULL`.	[allow-none]
stock_id	the stock icon to display in widgets representing the action, or `NULL`.	[allow-none]

Returns

the newly created GtkRecentAction.

Since: 2.12

---

gtk_recent_action_new_for_manager ()

```
GtkAction *
gtk_recent_action_new_for_manager (const gchar *name,
                                   const gchar *label,
                                   const gchar *tooltip,
                                   const gchar *stock_id,
                                   GtkRecentManager *manager);
```

`gtk_recent_action_new_for_manager` has been deprecated since version 3.10 and should not be used in newly-written code.

Creates a new GtkRecentAction object. To add the action to a GtkActionGroup and set the accelerator for the action, call `gtk_action_group_add_action_with_accel()`.

Parameters

name	a unique name for the action	
label	the label displayed in menu items and on buttons, or `NULL`.	[allow-none]
tooltip	a tooltip for the action, or `NULL`.	[allow-none]
stock_id	the stock icon to display in widgets representing the action, or `NULL`.	[allow-none]
manager	a GtkRecentManager, or `NULL` for using the default GtkRecentManager.	[allow-none]

Returns

the newly created GtkRecentAction

Since: 2.12

---

gtk_recent_action_get_show_numbers ()

```
gboolean
gtk_recent_action_get_show_numbers (GtkRecentAction *action);
```

`gtk_recent_action_get_show_numbers` has been deprecated since version 3.10 and should not be used in newly-written code.

Returns the value set by `gtk_recent_chooser_menu_set_show_numbers()`.

Parameters

action	a GtkRecentAction

Returns

`TRUE` if numbers should be shown.

Since: 2.12

---

gtk_recent_action_set_show_numbers ()

```
void
gtk_recent_action_set_show_numbers (GtkRecentAction *action,
                                    gboolean show_numbers);
```

`gtk_recent_action_set_show_numbers` has been deprecated since version 3.10 and should not be used in newly-written code.

Sets whether a number should be added to the items shown by the widgets representing `action` . The numbers are shown to provide a unique character for a mnemonic to be used inside the menu item's label. Only the first ten items get a number to avoid clashes.

Parameters

action	a GtkRecentAction
show_numbers	`TRUE` if the shown items should be numbered

Since: 2.12

Types and Values

struct GtkRecentAction

```
struct GtkRecentAction;
```

Property Details

The “show-numbers” property

“show-numbers” gboolean

Whether the items should be displayed with a number.

GtkRecentAction:show-numbers has been deprecated since version 3.10 and should not be used in newly-written code.

Flags: Read / Write

Default value: FALSE

GtkActivatable

GtkActivatable — An interface for activatable widgets

Functions

void	gtk_activatable_do_set_related_action ()
GtkAction *	gtk_activatable_get_related_action ()
gboolean	gtk_activatable_get_use_action_appearance ()
void	gtk_activatable_sync_action_properties ()
void	gtk_activatable_set_related_action ()
void	gtk_activatable_set_use_action_appearance ()

Properties

GtkAction *	related-action	Read / Write
gboolean	use-action-appearance	Read / Write

Types and Values

	GtkActivatable
struct	GtkActivatableIface

Object Hierarchy

```
GInterface
└── GtkActivatable
```

Prerequisites

GtkActivatable requires GObject.

Known Implementations

GtkActivatable is implemented by GtkButton, GtkCheckButton, GtkCheckMenuItem, GtkColorButton, GtkFontButton, GtkImageMenuItem, GtkLinkButton, GtkLockButton, GtkMenuButton, GtkMenuItem,

GtkMenuToolButton, GtkModelButton, GtkRadioButton, GtkRadioMenuItem, GtkRadioToolButton, GtkRecentChooserMenu, GtkScaleButton, GtkSeparatorMenuItem, GtkSeparatorToolItem, GtkSwitch, GtkTearoffMenuItem, GtkToggleButton, GtkToggleToolButton, GtkToolButton, GtkToolItem and GtkVolumeButton.

Includes

```
#include <gtk/gtk.h>
```

Description

Activatable widgets can be connected to a GtkAction and reflects the state of its action. A GtkActivatable can also provide feedback through its action, as they are responsible for activating their related actions.

Implementing GtkActivatable

When extending a class that is already GtkActivatable; it is only necessary to implement the GtkActivatable->`sync_action_properties()` and GtkActivatable->`update()` methods and chain up to the parent implementation, however when introducing a new GtkActivatable class; the "related-action" and "use-action-appearance" properties need to be handled by the implementor. Handling these properties is mostly a matter of installing the action pointer and boolean flag on your instance, and calling `gtk_activatable_do_set_related_action()` and `gtk_activatable_sync_action_properties()` at the appropriate times.

A class fragment implementing GtkActivatable

```
enum {
...

PROP_ACTIVATABLE_RELATED_ACTION,
PROP_ACTIVATABLE_USE_ACTION_APPEARANCE
}

struct _FooBarPrivate
{

  ...

  GtkAction      *action;
  gboolean        use_action_appearance;
};

...

static void
foo_bar_activatable_interface_init
(GtkActivatableIface  *iface);
static void foo_bar_activatable_update
(GtkActivatable       *activatable,

GtkAction            *action,

const gchar          *property_name);
```

```
static void
foo_bar_activatable_sync_action_properties
(GtkActivatable       *activatable,

GtkAction            *action);
...


static void
foo_bar_class_init (FooBarClass *klass)
{

  ...

  g_object_class_override_property
(gobject_class,
PROP_ACTIVATABLE_RELATED_ACTION, "related-
action");
  g_object_class_override_property
(gobject_class,
PROP_ACTIVATABLE_USE_ACTION_APPEARANCE, "use-
action-appearance");

  ...
}


static void
foo_bar_activatable_interface_init
(GtkActivatableIface  *iface)
{
  iface->update = foo_bar_activatable_update;
  iface->sync_action_properties =
foo_bar_activatable_sync_action_properties;
}

... Break the reference using
gtk_activatable_do_set_related_action()...

static void
foo_bar_dispose (GObject *object)
{
  FooBar *bar = FOO_BAR (object);
  FooBarPrivate *priv = FOO_BAR_GET_PRIVATE
(bar);

  ...

  if (priv->action)
    {
      gtk_activatable_do_set_related_action
(GTK_ACTIVATABLE (bar), NULL);
      priv->action = NULL;
    }
  G_OBJECT_CLASS (foo_bar_parent_class)-
>dispose (object);
}

... Handle the “related-action” and “use-
action-appearance” properties ...

static void
foo_bar_set_property (GObject
```

```
*object,
                        guint
prop_id,
                        const GValue    *value,
                        GParamSpec      *pspec)
{
  FooBar *bar = FOO_BAR (object);
  FooBarPrivate *priv = FOO_BAR_GET_PRIVATE
(bar);

  switch (prop_id)
    {

      ...

    case PROP_ACTIVATABLE_RELATED_ACTION:
      foo_bar_set_related_action (bar,
g_value_get_object (value));
      break;
    case
PROP_ACTIVATABLE_USE_ACTION_APPEARANCE:
      foo_bar_set_use_action_appearance (bar,
g_value_get_boolean (value));
      break;
    default:
      G_OBJECT_WARN_INVALID_PROPERTY_ID
(object, prop_id, pspec);
      break;
    }
}

static void
foo_bar_get_property (GObject
*object,
                            guint
prop_id,
                            GValue
*value,
                            GParamSpec
*pspec)
{
  FooBar *bar = FOO_BAR (object);
  FooBarPrivate *priv = FOO_BAR_GET_PRIVATE
(bar);

  switch (prop_id)
    {

      ...

    case PROP_ACTIVATABLE_RELATED_ACTION:
      g_value_set_object (value, priv-
>action);
      break;
    case
PROP_ACTIVATABLE_USE_ACTION_APPEARANCE:
      g_value_set_boolean (value, priv-
>use_action_appearance);
      break;
    default:
      G_OBJECT_WARN_INVALID_PROPERTY_ID
(object, prop_id, pspec);
      break;
```

```
    }
}


static void
foo_bar_set_use_action_appearance (FooBar
*bar,
                                   gboolean
use_appearance)
{
  FooBarPrivate *priv = FOO_BAR_GET_PRIVATE
(bar);

  if (priv->use_action_appearance !=
use_appearance)
    {
      priv->use_action_appearance =
use_appearance;

      gtk_activatable_sync_action_properties
(GTK_ACTIVATABLE (bar), priv->action);
    }
}

... call
gtk_activatable_do_set_related_action() and
then assign the action pointer,
no need to reference the action here since
gtk_activatable_do_set_related_action()
already
holds a reference here for you...
static void
foo_bar_set_related_action (FooBar    *bar,
                            GtkAction *action)
{
  FooBarPrivate *priv = FOO_BAR_GET_PRIVATE
(bar);

  if (priv->action == action)
    return;

  gtk_activatable_do_set_related_action
(GTK_ACTIVATABLE (bar), action);

  priv->action = action;
}

... Selectively reset and update activatable
depending on the use-action-appearance
property ...
static void
gtk_button_activatable_sync_action_properties
(GtkActivatable       *activatable,

GtkAction            *action)
{
  GtkButtonPrivate *priv =
GTK_BUTTON_GET_PRIVATE (activatable);

  if (!action)
    return;

  if (gtk_action_is_visible (action))
```

```
    gtk_widget_show (GTK_WIDGET
(activatable));
  else
    gtk_widget_hide (GTK_WIDGET
(activatable));

  gtk_widget_set_sensitive (GTK_WIDGET
(activatable), gtk_action_is_sensitive
(action));

  ...

  if (priv->use_action_appearance)
    {
      if (gtk_action_get_stock_id (action))
      foo_bar_set_stock (button,
gtk_action_get_stock_id (action));
      else if (gtk_action_get_label (action))
      foo_bar_set_label (button,
gtk_action_get_label (action));

      ...

    }
}

static void
foo_bar_activatable_update (GtkActivatable
*activatable,
                            GtkAction
*action,
                            const gchar
*property_name)
{
  FooBarPrivate *priv = FOO_BAR_GET_PRIVATE
(activatable);

  if (strcmp (property_name, "visible") == 0)
    {
      if (gtk_action_is_visible (action))
      gtk_widget_show (GTK_WIDGET
(activatable));
      else
      gtk_widget_hide (GTK_WIDGET
(activatable));
    }
  else if (strcmp (property_name,
"sensitive") == 0)
    gtk_widget_set_sensitive (GTK_WIDGET
(activatable), gtk_action_is_sensitive
(action));

  ...

  if (!priv->use_action_appearance)
    return;

  if (strcmp (property_name, "stock-id") ==
0)
    foo_bar_set_stock (button,
gtk_action_get_stock_id (action));
  else if (strcmp (property_name, "label") ==
0)
```

```
    foo_bar_set_label (button,
gtk_action_get_label (action));

  ...
}
```

Functions

gtk_activatable_do_set_related_action ()

```
void
gtk_activatable_do_set_related_action (GtkActivatable *activatable,
                                       GtkAction *action);
```

`gtk_activatable_do_set_related_action` has been deprecated since version 3.10 and should not be used in newly-written code.

This is a utility function for GtkActivatable implementors.

When implementing GtkActivatable you must call this when handling changes of the "related-action", and you must also use this to break references in GObject->`dispose()`.

This function adds a reference to the currently set related action for you, it also makes sure the GtkActivatable->`update()` method is called when the related GtkAction properties change and registers to the action's proxy list.

> Be careful to call this before setting the local copy of the GtkAction property, since this function uses `gtk_activatable_get_related_action()` to retrieve the previous action.

Parameters

activatable	a GtkActivatable
action	the GtkAction to set

Since: 2.16

---

gtk_activatable_get_related_action ()

```
GtkAction *
gtk_activatable_get_related_action (GtkActivatable *activatable);
```

`gtk_activatable_get_related_action` has been deprecated since version 3.10 and should not be used in newly-written code.

Gets the related GtkAction for `activatable` .

Parameters

activatable	a GtkActivatable

Returns

the related GtkAction if one is set.

[transfer none]

Since: 2.16

---

gtk_activatable_get_use_action_appearance ()

```
gboolean
gtk_activatable_get_use_action_appearance
                                (GtkActivatable *activatable);
```

`gtk_activatable_get_use_action_appearance` has been deprecated since version 3.10 and should not be used in newly-written code.

Gets whether this activatable should reset its layout and appearance when setting the related action or when the action changes appearance.

Parameters

activatable	a GtkActivatable

Returns

whether `activatable` uses its actions appearance.

Since: 2.16

---

gtk_activatable_sync_action_properties ()

```
void
gtk_activatable_sync_action_properties
                                (GtkActivatable *activatable,
                                 GtkAction *action);
```

`gtk_activatable_sync_action_properties` has been deprecated since version 3.10 and should not be used in newly-written code.

This is called to update the activatable completely, this is called internally when the “related-action” property is set or unset and by the implementing class when “use-action-appearance” changes.

Parameters

activatable	a GtkActivatable	
action	the related GtkAction or `NULL`.	[allow-none]

Since: 2.16

---

gtk_activatable_set_related_action ()

```
void
gtk_activatable_set_related_action (GtkActivatable *activatable,
                                    GtkAction *action);
```

`gtk_activatable_set_related_action` has been deprecated since version 3.10 and should not be used in newly-written code.

Sets the related action on the `activatable` object.

> GtkActivatable implementors need to handle the "related-action" property and call `gtk_activatable_do_set_related_action()` when it changes.

Parameters

activatable	a GtkActivatable
action	the GtkAction to set

Since: 2.16

---

gtk_activatable_set_use_action_appearance ()

```
void
gtk_activatable_set_use_action_appearance
                                (GtkActivatable *activatable,
                                 gboolean use_appearance);
```

`gtk_activatable_set_use_action_appearance` has been deprecated since version 3.10 and should not be used in newly-written code.

Sets whether this activatable should reset its layout and appearance when setting the related action or when the action changes appearance

> GtkActivatable implementors need to handle the "use-action-appearance" property and call `gtk_activatable_sync_action_properties()` to update `activatable` if needed.

Parameters

activatable	a GtkActivatable
use_appearance	whether to use the actions appearance

Since: 2.16

Types and Values

GtkActivatable

```
typedef struct _GtkActivatable GtkActivatable;
```

---

struct GtkActivatableIface

```
struct GtkActivatableIface {
  /* virtual table */
  void   (* update)                 (GtkActivatable *activatable,
                                     GtkAction      *action,
                                     const gchar    *property_name);
  void   (* sync_action_properties)   (GtkActivatable *activatable,
                                     GtkAction      *action);
};
```

`GtkActivatableIface` has been deprecated since version 3.10 and should not be used in newly-written code.

This method can be called with a `NULL` action at times.

Members

`update` ()	Called to update the activatable when its related action's properties change. You must check the "use-action-appearance" property only apply action properties that are meant to effect the appearance accordingly.
`sync_action_properties` ()	Called to update the activatable completely, this is called internally when "related-action" property is set or unset and by the implementor when "use-action-appearance" changes.

Since: 2.16

Property Details

The `"related-action"` property

```
  "related-action"           GtkAction *
```

The action that this activatable will activate and receive updates from for various states and possibly appearance.

GtkActivatable implementors need to handle the this property and call `gtk_activatable_do_set_related_action()` when it changes.

`GtkActivatable:related-action` has been deprecated since version 3.10 and should not be used in newly-written code.

Flags: Read / Write

Since: 2.16

The “use-action-appearance” property

“use-action-appearance” gboolean

Whether this activatable should reset its layout and appearance when setting the related action or when the action changes appearance.

See the GtkAction documentation directly to find which properties should be ignored by the GtkActivatable when this property is FALSE.

> GtkActivatable implementors need to handle this property and call gtk_activatable_sync_action_properties() on the activatable widget when it changes.

GtkActivatable:use-action-appearance has been deprecated since version 3.10 and should not be used in newly-written code.

Flags: Read / Write

Default value: TRUE

Since: 2.16

GtkImageMenuItem

GtkImageMenuItem — A deprecated widget for a menu item with an icon

Functions

void	gtk_image_menu_item_set_image ()
GtkWidget *	gtk_image_menu_item_get_image ()
GtkWidget *	gtk_image_menu_item_new ()
GtkWidget *	gtk_image_menu_item_new_from_stock ()
GtkWidget *	gtk_image_menu_item_new_with_label ()
GtkWidget *	gtk_image_menu_item_new_with_mnemonic ()
gboolean	gtk_image_menu_item_get_use_stock ()
void	gtk_image_menu_item_set_use_stock ()
gboolean	gtk_image_menu_item_get_always_show_image ()
void	gtk_image_menu_item_set_always_show_image ()
void	gtk_image_menu_item_set_accel_group ()

Properties

GtkAccelGroup *	accel-group	Write
gboolean	always-show-image	Read / Write / Construct
GtkWidget *	image	Read / Write
gboolean	use-stock	Read / Write / Construct

Types and Values

struct	GtkImageMenuItem
struct	GtkImageMenuItemClass

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkContainer
            └── GtkBin
                └── GtkMenuItem
                    └── GtkImageMenuItem
```

Implemented Interfaces

GtkImageMenuItem implements AtkImplementorIface, GtkBuildable, GtkActivatable and GtkActionable.

Includes

```
#include <gtk/gtk.h>
```

Description

A GtkImageMenuItem is a menu item which has an icon next to the text label.

This is functionally equivalent to:

```
GtkWidget *box = gtk_box_new
(GTK_ORIENTATION_HORIZONTAL, 6);
GtkWidget *icon =
gtk_image_new_from_icon_name ("folder-music-
symbolic", GTK_ICON_SIZE_MENU);
GtkWidget *label = gtk_label_new ("Music");
GtkWidget *menu_item = gtk_menu_item_new ();

gtk_container_add (GTK_CONTAINER (box),
icon);
gtk_container_add (GTK_CONTAINER (box),
label);

gtk_container_add (GTK_CONTAINER (menu_item),
box);

gtk_widget_show_all (menu_item);
```

Note that the user may disable display of menu icons using the "gtk-menu-images" setting, so make sure to still fill in the text label. If you want to ensure that your menu items show an icon you are strongly encouraged to use a GtkMenuItem with a GtkImage instead.

GtkImageMenuItem has been deprecated since GTK+ 3.10. If you want to display an icon in a menu item, you should use GtkMenuItem and pack a GtkBox with a GtkImage and a GtkLabel instead. You should also consider using GtkBuilder and the XML GMenu description for creating menus, by following the GMenu guide. You should consider using icons in menu items only sparingly, and for "objects" (or "nouns") elements only, like bookmarks, files, and links; "actions" (or "verbs") should not have icons.

Furthermore, if you would like to display keyboard accelerator, you must pack the accel label into the box using `gtk_box_pack_end()` and align the label, otherwise the accelerator will not display correctly. The following code snippet adds a keyboard accelerator to the menu item, with a key binding of Ctrl+M:

```
GtkWidget *box = gtk_box_new
```

```
2                                                   (GTK_ORIENTATION_HORIZONTAL, 6);
3                                                   GtkWidget *icon =
4                                                   gtk_image_new_from_icon_name ("folder-music-
5                                                   symbolic", GTK_ICON_SIZE_MENU);
6                                                   GtkWidget *label = gtk_accel_label_new
7                                                   ("Music");
8                                                   GtkWidget *menu_item = gtk_menu_item_new ();
9                                                   GtkAccelGroup *accel_group =
10                                                  gtk_accel_group_new ();
11
12                                                  gtk_container_add (GTK_CONTAINER (box),
13                                                  icon);
14
15                                                  gtk_label_set_use_underline (GTK_LABEL
16                                                  (label), TRUE);
17                                                  gtk_label_set_xalign (GTK_LABEL (label),
18                                                  0.0);
19
20                                                  gtk_widget_add_accelerator (menu_item,
                                                    "activate", accel_group,
                                                                                GDK_KEY_m,
                                                    GDK_CONTROL_MASK, GTK_ACCEL_VISIBLE);
                                                    gtk_accel_label_set_accel_widget
                                                    (GTK_ACCEL_LABEL (label), menu_item);

                                                    gtk_box_pack_end (GTK_BOX (box), label, TRUE,
                                                    TRUE, 0);

                                                    gtk_container_add (GTK_CONTAINER (menu_item),
                                                    box);

                                                    gtk_widget_show_all (menu_item);
```

Functions

gtk_image_menu_item_set_image ()

```
void
gtk_image_menu_item_set_image (GtkImageMenuItem *image_menu_item,
                               GtkWidget *image);
```

`gtk_image_menu_item_set_image` has been deprecated since version 3.10 and should not be used in newly-written code.

Sets the image of `image_menu_item` to the given widget. Note that it depends on the show-menu-images setting whether the image will be displayed or not.

Parameters

image_menu_item	a GtkImageMenuItem.	
image	a widget to set as the image for the menu item.	[allow-none]

gtk_image_menu_item_get_image ()

```
GtkWidget *
gtk_image_menu_item_get_image (GtkImageMenuItem *image_menu_item);
```

`gtk_image_menu_item_get_image` has been deprecated since version 3.10 and should not be used in newly-written code.

Gets the widget that is currently set as the image of `image_menu_item` . See `gtk_image_menu_item_set_image()`.

Parameters

image_menu_item	a GtkImageMenuItem

Returns

the widget set as image of `image_menu_item` .

[transfer none]

---

gtk_image_menu_item_new ()

```
GtkWidget *
gtk_image_menu_item_new (void);
```

`gtk_image_menu_item_new` has been deprecated since version 3.10 and should not be used in newly-written code.

Use `gtk_menu_item_new()` instead.

Creates a new GtkImageMenuItem with an empty label.

Returns

a new GtkImageMenuItem

---

gtk_image_menu_item_new_from_stock ()

```
GtkWidget *
gtk_image_menu_item_new_from_stock (const gchar *stock_id,
                                    GtkAccelGroup *accel_group);
```

`gtk_image_menu_item_new_from_stock` has been deprecated since version 3.10 and should not be used in newly-written code.

Use `gtk_menu_item_new_with_mnemonic()` instead.

Creates a new GtkImageMenuItem containing the image and text from a stock item. Some stock ids have preprocessor macros like GTK_STOCK_OK and GTK_STOCK_APPLY.

If you want this menu item to have changeable accelerators, then pass in `NULL` for accel_group. Next call `gtk_menu_item_set_accel_path()` with an appropriate path for the menu item, use `gtk_stock_lookup()` to

look up the standard accelerator for the stock item, and if one is found, call `gtk_accel_map_add_entry()` to register it.

Parameters

stock_id	the name of the stock item.
accel_group	the GtkAccelGroup to add the menu [allow-none] items accelerator to, or `NULL`.

Returns

a new GtkImageMenuItem.

---

gtk_image_menu_item_new_with_label ()

```
GtkWidget *
gtk_image_menu_item_new_with_label (const gchar *label);
```

`gtk_image_menu_item_new_with_label` has been deprecated since version 3.10 and should not be used in newly-written code.

Use `gtk_menu_item_new_with_label()` instead.

Creates a new GtkImageMenuItem containing a label.

Parameters

label	the text of the menu item.

Returns

a new GtkImageMenuItem.

---

gtk_image_menu_item_new_with_mnemonic ()

```
GtkWidget *
gtk_image_menu_item_new_with_mnemonic (const gchar *label);
```

`gtk_image_menu_item_new_with_mnemonic` has been deprecated since version 3.10 and should not be used in newly-written code.

Use `gtk_menu_item_new_with_mnemonic()` instead.

Creates a new GtkImageMenuItem containing a label. The label will be created using `gtk_label_new_with_mnemonic()`, so underscores in `label` indicate the mnemonic for the menu item.

Parameters

label	the text of the menu item, with an underscore in front of the mnemonic character

Returns

a new GtkImageMenuItem

---

gtk_image_menu_item_get_use_stock ()

```
gboolean
gtk_image_menu_item_get_use_stock (GtkImageMenuItem *image_menu_item);
```

`gtk_image_menu_item_get_use_stock` has been deprecated since version 3.10 and should not be used in newly-written code.

Checks whether the label set in the menuitem is used as a stock id to select the stock item for the item.

Parameters

image_menu_item	a GtkImageMenuItem

Returns

`TRUE` if the label set in the menuitem is used as a stock id to select the stock item for the item

Since: 2.16

---

gtk_image_menu_item_set_use_stock ()

```
void
gtk_image_menu_item_set_use_stock (GtkImageMenuItem *image_menu_item,
                                   gboolean use_stock);
```

`gtk_image_menu_item_set_use_stock` has been deprecated since version 3.10 and should not be used in newly-written code.

If `TRUE`, the label set in the menuitem is used as a stock id to select the stock item for the item.

Parameters

image_menu_item	a GtkImageMenuItem
use_stock	`TRUE` if the menuitem should use a stock item

Since: 2.16

---

gtk_image_menu_item_get_always_show_image ()

```
gboolean
gtk_image_menu_item_get_always_show_image
                                (GtkImageMenuItem *image_menu_item);
```

`gtk_image_menu_item_get_always_show_image` has been deprecated since version 3.10 and should not be used in newly-written code.

Returns whether the menu item will ignore the "gtk-menu-images" setting and always show the image, if available.

Parameters

image_menu_item	a GtkImageMenuItem

Returns

`TRUE` if the menu item will always show the image

Since: 2.16

---

gtk_image_menu_item_set_always_show_image ()

```
void
gtk_image_menu_item_set_always_show_image
                                (GtkImageMenuItem *image_menu_item,
                                 gboolean always_show);
```

`gtk_image_menu_item_set_always_show_image` has been deprecated since version 3.10 and should not be used in newly-written code.

If `TRUE`, the menu item will ignore the "gtk-menu-images" setting and always show the image, if available.

Use this property if the menuitem would be useless or hard to use without the image.

Parameters

image_menu_item	a GtkImageMenuItem
always_show	`TRUE` if the menuitem should always show the image

Since: 2.16

---

gtk_image_menu_item_set_accel_group ()

```
void
gtk_image_menu_item_set_accel_group (GtkImageMenuItem *image_menu_item,
                                     GtkAccelGroup *accel_group);
```

`gtk_image_menu_item_set_accel_group` has been deprecated since version 3.10 and should not be used in newly-written code.

Specifies an `accel_group` to add the menu items accelerator to (this only applies to stock items so a stock item

must already be set, make sure to call gtk_image_menu_item_set_use_stock() and gtk_menu_item_set_label() with a valid stock item first).

If you want this menu item to have changeable accelerators then you shouldnt need this (see gtk_image_menu_item_new_from_stock()).

Parameters

image_menu_item	a GtkImageMenuItem
accel_group	the GtkAccelGroup

Since: 2.16

Types and Values

struct GtkImageMenuItem

```
struct GtkImageMenuItem;
```

---

struct GtkImageMenuItemClass

```
struct GtkImageMenuItemClass {
  GtkMenuItemClass parent_class;
};
```

Members

Property Details

The “accel-group” property

```
  “accel-group”              GtkAccelGroup *
```

The Accel Group to use for stock accelerator keys

`GtkImageMenuItem:accel-group` has been deprecated since version 3.10 and should not be used in newly-written code.

Use gtk_widget_add_accelerator() instead

Flags: Write

Since: 2.16

The “always-show-image” property

“always-show-image” gboolean

If TRUE, the menu item will always show the image, if available.

Use this property only if the menuitem would be useless or hard to use without the image.

GtkImageMenuItem:always-show-image has been deprecated since version 3.10 and should not be used in newly-written code.

Use a GtkMenuItem containing a GtkBox with a GtkAccelLabel and a GtkImage instead

Flags: Read / Write / Construct

Default value: FALSE

Since: 2.16

The “image” property

“image” GtkWidget *

Child widget to appear next to the menu text.

GtkImageMenuItem:image has been deprecated since version 3.10 and should not be used in newly-written code.

Use a GtkMenuItem containing a GtkBox with a GtkAccelLabel and a GtkImage instead

Flags: Read / Write

The “use-stock” property

“use-stock” gboolean

If TRUE, the label set in the menuitem is used as a stock id to select the stock item for the item.

GtkImageMenuItem:use-stock has been deprecated since version 3.10 and should not be used in newly-written code.

Use a named icon from the GtkIconTheme instead

Flags: Read / Write / Construct

Default value: FALSE

Since: 2.16

GtkMisc

GtkMisc — Base class for widgets with alignments and padding

Functions

void	gtk_misc_set_alignment ()
void	gtk_misc_set_padding ()
void	gtk_misc_get_alignment ()
void	gtk_misc_get_padding ()

Properties

gfloat	xalign	Read / Write
gint	xpad	Read / Write
gfloat	yalign	Read / Write
gint	ypad	Read / Write

Types and Values

struct	GtkMisc

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkMisc
            ├── GtkLabel
            ├── GtkArrow
            └── GtkImage
```

Implemented Interfaces

GtkMisc implements AtkImplementorIface and GtkBuildable.

Includes

```
#include <gtk/gtk.h>
```

Description

The GtkMisc widget is an abstract widget which is not useful itself, but is used to derive subclasses which have alignment and padding attributes.

The horizontal and vertical padding attributes allows extra space to be added around the widget.

The horizontal and vertical alignment attributes enable the widget to be positioned within its allocated area. Note that if the widget is added to a container in such a way that it expands automatically to fill its allocated area, the alignment settings will not alter the widget's position.

Note that the desired effect can in most cases be achieved by using the “halign”, “valign” and “margin” properties on the child widget, so GtkMisc should not be used in new code. To reflect this fact, all GtkMisc API has been deprecated.

Functions

gtk_misc_set_alignment ()

```
void
gtk_misc_set_alignment (GtkMisc *misc,
                        gfloat xalign,
                        gfloat yalign);
```

`gtk_misc_set_alignment` has been deprecated since version 3.14 and should not be used in newly-written code.

Use GtkWidget's alignment (“halign” and “valign”) and margin properties or GtkLabel's “xalign” and “yalign” properties.

Sets the alignment of the widget.

Parameters

misc	a GtkMisc.
xalign	the horizontal alignment, from 0 (left) to 1 (right).
yalign	the vertical alignment, from 0 (top) to 1 (bottom).

---

gtk_misc_set_padding ()

```
void
gtk_misc_set_padding (GtkMisc *misc,
                      gint xpad,
                      gint ypad);
```

`gtk_misc_set_padding` has been deprecated since version 3.14 and should not be used in newly-written code.

Use GtkWidget alignment and margin properties.

Sets the amount of space to add around the widget.

Parameters

misc	a GtkMisc.
xpad	the amount of space to add on the left and right of the widget, in pixels.
ypad	the amount of space to add on the top and bottom of the widget, in pixels.

---

gtk_misc_get_alignment ()

```
void
gtk_misc_get_alignment (GtkMisc *misc,
                        gfloat *xalign,
                        gfloat *yalign);
```

`gtk_misc_get_alignment` has been deprecated since version 3.14 and should not be used in newly-written code.

Use GtkWidget alignment and margin properties.

Gets the X and Y alignment of the widget within its allocation. See `gtk_misc_set_alignment()`.

Parameters

misc	a GtkMisc	
xalign	location to store X alignment of `misc` , or `NULL`.	[out][allow-none]
yalign	location to store Y alignment of `misc` , or `NULL`.	[out][allow-none]

---

gtk_misc_get_padding ()

```
void
gtk_misc_get_padding (GtkMisc *misc,
                      gint *xpad,
                      gint *ypad);
```

`gtk_misc_get_padding` has been deprecated since version 3.14 and should not be used in newly-written code.

Use GtkWidget alignment and margin properties.

Gets the padding in the X and Y directions of the widget. See `gtk_misc_set_padding()`.

Parameters

misc	a GtkMisc	
xpad	location to store padding in the X direction, or `NULL`.	[out][allow-none]
ypad	location to store padding in the Y direction, or `NULL`.	[out][allow-none]

Types and Values

struct GtkMisc

```
struct GtkMisc;
```

Property Details

The “xalign” property

“xalign” gfloat

The horizontal alignment. A value of 0.0 means left alignment (or right on RTL locales); a value of 1.0 means right alignment (or left on RTL locales).

GtkMisc:xalign has been deprecated since version 3.14 and should not be used in newly-written code.

Use gtk_widget_set_halign() instead. If you are using GtkLabel, use “xalign” instead.

Flags: Read / Write

Allowed values: [0,1]

Default value: 0.5

The “xpad” property

“xpad” gint

The amount of space to add on the left and right of the widget, in pixels.

GtkMisc:xpad has been deprecated since version 3.14 and should not be used in newly-written code.

Use gtk_widget_set_margin_start() and gtk_widget_set_margin_end() instead

Flags: Read / Write

Allowed values: >= 0

Default value: 0

The “yalign” property

“yalign” gfloat

The vertical alignment. A value of 0.0 means top alignment; a value of 1.0 means bottom alignment.

GtkMisc:yalign has been deprecated since version 3.14 and should not be used in newly-written code.

Use gtk_widget_set_valign() instead. If you are using GtkLabel, use “yalign” instead.

Flags: Read / Write

Allowed values: [0,1]

Default value: 0.5

The “ypad” property

“ypad” gint

The amount of space to add on the top and bottom of the widget, in pixels.

`GtkMisc:ypad` has been deprecated since version 3.14 and should not be used in newly-written code.

Use `gtk_widget_set_margin_top()` and `gtk_widget_set_margin_bottom()` instead

Flags: Read / Write

Allowed values: >= 0

Default value: 0

Stock Items

Stock Items — Prebuilt common menu/toolbar items and corresponding icons

Functions

void	gtk_stock_add ()
void	gtk_stock_add_static ()
GtkStockItem *	gtk_stock_item_copy ()
void	gtk_stock_item_free ()
GSList *	gtk_stock_list_ids ()
gboolean	gtk_stock_lookup ()
void	gtk_stock_set_translate_func ()

Types and Values

struct	GtkStockItem
#define	GTK_STOCK_ABOUT
#define	GTK_STOCK_ADD
#define	GTK_STOCK_APPLY
#define	GTK_STOCK_BOLD
#define	GTK_STOCK_CANCEL
#define	GTK_STOCK_CAPS_LOCK_WARNING
#define	GTK_STOCK_CDROM
#define	GTK_STOCK_CLEAR
#define	GTK_STOCK_CLOSE
#define	GTK_STOCK_COLOR_PICKER
#define	GTK_STOCK_CONVERT
#define	GTK_STOCK_CONNECT
#define	GTK_STOCK_COPY
#define	GTK_STOCK_CUT
#define	GTK_STOCK_DELETE
#define	GTK_STOCK_DIALOG_AUTHENTICATION
#define	GTK_STOCK_DIALOG_ERROR
#define	GTK_STOCK_DIALOG_INFO
#define	GTK_STOCK_DIALOG_QUESTION
#define	GTK_STOCK_DIALOG_WARNING

#define GTK_STOCK_DIRECTORY
#define GTK_STOCK_DISCARD
#define GTK_STOCK_DISCONNECT
#define GTK_STOCK_DND
#define GTK_STOCK_DND_MULTIPLE
#define GTK_STOCK_EDIT
#define GTK_STOCK_EXECUTE
#define GTK_STOCK_FILE
#define GTK_STOCK_FIND
#define GTK_STOCK_FIND_AND_REPLACE
#define GTK_STOCK_FLOPPY
#define GTK_STOCK_FULLSCREEN
#define GTK_STOCK_GOTO_BOTTOM
#define GTK_STOCK_GOTO_FIRST
#define GTK_STOCK_GOTO_LAST
#define GTK_STOCK_GOTO_TOP
#define GTK_STOCK_GO_BACK
#define GTK_STOCK_GO_DOWN
#define GTK_STOCK_GO_FORWARD
#define GTK_STOCK_GO_UP
#define GTK_STOCK_HARDDISK
#define GTK_STOCK_HELP
#define GTK_STOCK_HOME
#define GTK_STOCK_INDENT
#define GTK_STOCK_INDEX
#define GTK_STOCK_INFO
#define GTK_STOCK_ITALIC
#define GTK_STOCK_JUMP_TO
#define GTK_STOCK_JUSTIFY_CENTER
#define GTK_STOCK_JUSTIFY_FILL
#define GTK_STOCK_JUSTIFY_LEFT
#define GTK_STOCK_JUSTIFY_RIGHT
#define GTK_STOCK_LEAVE_FULLSCREEN
#define GTK_STOCK_MEDIA_FORWARD
#define GTK_STOCK_MEDIA_NEXT
#define GTK_STOCK_MEDIA_PAUSE
#define GTK_STOCK_MEDIA_PLAY
#define GTK_STOCK_MEDIA_PREVIOUS
#define GTK_STOCK_MEDIA_RECORD
#define GTK_STOCK_MEDIA_REWIND
#define GTK_STOCK_MEDIA_STOP
#define GTK_STOCK_MISSING_IMAGE
#define GTK_STOCK_NETWORK
#define GTK_STOCK_NEW
#define GTK_STOCK_NO
#define GTK_STOCK_OK
#define GTK_STOCK_OPEN
#define GTK_STOCK_ORIENTATION_LANDSCAPE
#define GTK_STOCK_ORIENTATION_PORTRAIT
#define GTK_STOCK_ORIENTATION_REVERSE_LANDSCAPE
#define GTK_STOCK_ORIENTATION_REVERSE_PORTR

AIT
#define GTK_STOCK_PAGE_SETUP
#define GTK_STOCK_PASTE
#define GTK_STOCK_PREFERENCES
#define GTK_STOCK_PRINT
#define GTK_STOCK_PRINT_ERROR
#define GTK_STOCK_PRINT_PAUSED
#define GTK_STOCK_PRINT_PREVIEW
#define GTK_STOCK_PRINT_REPORT
#define GTK_STOCK_PRINT_WARNING
#define GTK_STOCK_PROPERTIES
#define GTK_STOCK_QUIT
#define GTK_STOCK_REDO
#define GTK_STOCK_REFRESH
#define GTK_STOCK_REMOVE
#define GTK_STOCK_REVERT_TO_SAVED
#define GTK_STOCK_SAVE
#define GTK_STOCK_SAVE_AS
#define GTK_STOCK_SELECT_ALL
#define GTK_STOCK_SELECT_COLOR
#define GTK_STOCK_SELECT_FONT
#define GTK_STOCK_SORT_ASCENDING
#define GTK_STOCK_SORT_DESCENDING
#define GTK_STOCK_SPELL_CHECK
#define GTK_STOCK_STOP
#define GTK_STOCK_STRIKETHROUGH
#define GTK_STOCK_UNDELETE
#define GTK_STOCK_UNDERLINE
#define GTK_STOCK_UNDO
#define GTK_STOCK_UNINDENT
#define GTK_STOCK_YES
#define GTK_STOCK_ZOOM_100
#define GTK_STOCK_ZOOM_FIT
#define GTK_STOCK_ZOOM_IN
#define GTK_STOCK_ZOOM_OUT

Includes

```
#include <gtk/gtk.h>
```

Description

> Since GTK+ 3.10, stock items are deprecated. You should instead set up whatever labels and/or icons you need using normal widget API, rather than relying on GTK+ providing ready-made combinations of these.

Stock items represent commonly-used menu or toolbar items such as “Open” or “Exit”. Each stock item is identified by a stock ID; stock IDs are just strings, but macros such as GTK_STOCK_OPEN are provided to avoid typing mistakes in the strings. Applications can register their own stock items in addition to those built-in to GTK+.

Each stock ID can be associated with a GtkStockItem, which contains the user-visible label, keyboard

accelerator, and translation domain of the menu or toolbar item; and/or with an icon stored in a GtkIconFactory. The connection between a GtkStockItem and stock icons is purely conventional (by virtue of using the same stock ID); it's possible to register a stock item but no icon, and vice versa. Stock icons may have a RTL variant which gets used for right-to-left locales.

Functions

gtk_stock_add ()

```
void
gtk_stock_add (const GtkStockItem *items,
               guint n_items);
```

`gtk_stock_add` has been deprecated since version 3.10 and should not be used in newly-written code.

Registers each of the stock items in `items` . If an item already exists with the same stock ID as one of the `items` , the old item gets replaced. The stock items are copied, so GTK+ does not hold any pointer into `items` and `items` can be freed. Use `gtk_stock_add_static()` if `items` is persistent and GTK+ need not copy the array.

Parameters

items	a GtkStockItem or array of items.	[array length=n_items]
n_items	number of GtkStockItem in `items`	

---

gtk_stock_add_static ()

```
void
gtk_stock_add_static (const GtkStockItem *items,
                      guint n_items);
```

`gtk_stock_add_static` has been deprecated since version 3.10 and should not be used in newly-written code.

Same as `gtk_stock_add()`, but doesn't copy `items` , so `items` must persist until application exit.

Parameters

items	a GtkStockItem or array of GtkStockItem.	[array length=n_items]
n_items	number of items	

---

gtk_stock_item_copy ()

```
GtkStockItem *
gtk_stock_item_copy (const GtkStockItem *item);
```

`gtk_stock_item_copy` has been deprecated since version 3.10 and should not be used in newly-written code.

Copies a stock item, mostly useful for language bindings and not in applications.

[skip]

Parameters

item	a GtkStockItem

Returns

a new GtkStockItem

---

gtk_stock_item_free ()

```
void
gtk_stock_item_free (GtkStockItem *item);
```

`gtk_stock_item_free` has been deprecated since version 3.10 and should not be used in newly-written code.

Frees a stock item allocated on the heap, such as one returned by `gtk_stock_item_copy()`. Also frees the fields inside the stock item, if they are not `NULL`.

Parameters

item	a GtkStockItem

---

gtk_stock_list_ids ()

```
GSList *
gtk_stock_list_ids (void);
```

`gtk_stock_list_ids` has been deprecated since version 3.10 and should not be used in newly-written code.

Retrieves a list of all known stock IDs added to a GtkIconFactory or registered with `gtk_stock_add()`. The list must be freed with `g_slist_free()`, and each string in the list must be freed with `g_free()`.

Returns

a list of known stock IDs.

[element-type utf8][transfer full]

---

gtk_stock_lookup ()

```
gboolean
gtk_stock_lookup (const gchar *stock_id,
                  GtkStockItem *item);
```

`gtk_stock_lookup` has been deprecated since version 3.10 and should not be used in newly-written code.

Fills `item` with the registered values for `stock_id` , returning `TRUE` if `stock_id` was known.

Parameters

stock_id	a stock item name	
item	stock item to initialize with values.	[out]

Returns

`TRUE` if `item` was initialized

---

gtk_stock_set_translate_func ()

```
void
gtk_stock_set_translate_func (const gchar *domain,
                              GtkTranslateFunc func,
                              gpointer data,
                              GDestroyNotify notify);
```

`gtk_stock_set_translate_func` has been deprecated since version 3.10 and should not be used in newly-written code.

Sets a function to be used for translating the `label` of a stock item.

If no function is registered for a translation domain, `g_dgettext()` is used.

The function is used for all stock items whose `translation_domain` matches `domain` . Note that it is possible to use strings different from the actual gettext translation domain of your application for this, as long as your GtkTranslateFunc uses the correct domain when calling `dgettext()`. This can be useful, e.g. when dealing with message contexts:

```
GtkStockItem items[] = {
 { MY_ITEM1, NC_("odd items", "Item 1"), 0,
0, "odd-item-domain" },
 { MY_ITEM2, NC_("even items", "Item 2"), 0,
0, "even-item-domain" },
};

gchar *
my_translate_func (const gchar *msgid,
                   gpointer     data)
{
  gchar *msgctxt = data;

  return (gchar*)g_dpgettext2
(GETTEXT_PACKAGE, msgctxt, msgid);
}

...

gtk_stock_add (items, G_N_ELEMENTS (items));
gtk_stock_set_translate_func ("odd-item-
domain", my_translate_func, "odd items");
gtk_stock_set_translate_func ("even-item-
domain", my_translate_func, "even items");
```

Parameters

domain	the translation domain for which `func` shall be used
func	a GtkTranslateFunc
data	data to pass to `func`
notify	a GDestroyNotify that is called when `data` is no longer needed

Since: 2.8

Types and Values

struct GtkStockItem

```
struct GtkStockItem {
  gchar *stock_id;
  gchar *label;
  GdkModifierType modifier;
  guint keyval;
  gchar *translation_domain;
};
```

`GtkStockItem` has been deprecated since version 3.10 and should not be used in newly-written code.

Members

gchar *`stock_id`;	Identifier.
gchar *`label`;	User visible label.
GdkModifierType `modifier`;	Modifier type for keyboard accelerator
guint `keyval`;	Keyboard accelerator
gchar *`translation_domain`;	Translation domain of the menu or toolbar item

GTK_STOCK_ABOUT

```
#define GTK_STOCK_ABOUT             ((GtkStock)"gtk-about")
```

`GTK_STOCK_ABOUT` has been deprecated since version 3.10 and should not be used in newly-written code.

Use named icon "help-about" or the label "_About".

The “About” item.

Since: 2.6

GTK_STOCK_ADD

```
#define GTK_STOCK_ADD               ((GtkStock)"gtk-add")
```

GTK_STOCK_ADD has been deprecated since version 3.10 and should not be used in newly-written code.

Use named icon "list-add" or the label "_Add".

The “Add” item and icon.

GTK_STOCK_APPLY

```
#define GTK_STOCK_APPLY            ((GtkStock)"gtk-apply")
```

GTK_STOCK_APPLY has been deprecated since version 3.10 and should not be used in newly-written code.

Do not use an icon. Use label "_Apply".

The “Apply” item and icon.

GTK_STOCK_BOLD

```
#define GTK_STOCK_BOLD             ((GtkStock)"gtk-bold")
```

GTK_STOCK_BOLD has been deprecated since version 3.10 and should not be used in newly-written code.

Use named icon "format-text-bold".

The “Bold” item and icon.

GTK_STOCK_CANCEL

```
#define GTK_STOCK_CANCEL           ((GtkStock)"gtk-cancel")
```

GTK_STOCK_CANCEL has been deprecated since version 3.10 and should not be used in newly-written code.

Do not use an icon. Use label "_Cancel".

The “Cancel” item and icon.

GTK_STOCK_CAPS_LOCK_WARNING

```
#define GTK_STOCK_CAPS_LOCK_WARNING ((GtkStock)"gtk-caps-lock-warning")
```

GTK_STOCK_CAPS_LOCK_WARNING has been deprecated since version 3.10 and should not be used in newly-written code.

Use named icon "dialog-warning-symbolic".

The “Caps Lock Warning” icon.

Since: 2.16

GTK_STOCK_CDROM

```
#define GTK_STOCK_CDROM             ((GtkStock)"gtk-cdrom")
```

`GTK_STOCK_CDROM` has been deprecated since version 3.10 and should not be used in newly-written code.

Use named icon "media-optical".

The “CD-Rom” item and icon.

GTK_STOCK_CLEAR

```
#define GTK_STOCK_CLEAR             ((GtkStock)"gtk-clear")
```

`GTK_STOCK_CLEAR` has been deprecated since version 3.10 and should not be used in newly-written code.

Use named icon "edit-clear".

The “Clear” item and icon.

GTK_STOCK_CLOSE

```
#define GTK_STOCK_CLOSE             ((GtkStock)"gtk-close")
```

`GTK_STOCK_CLOSE` has been deprecated since version 3.10 and should not be used in newly-written code.

Use named icon "window-close" or the label "_Close".

The “Close” item and icon.

GTK_STOCK_COLOR_PICKER

```
#define GTK_STOCK_COLOR_PICKER      ((GtkStock)"gtk-color-picker")
```

`GTK_STOCK_COLOR_PICKER` has been deprecated since version 3.10 and should not be used in newly-written code.

The “Color Picker” item and icon.

Since: 2.2

GTK_STOCK_CONVERT

```
#define GTK_STOCK_CONVERT           ((GtkStock)"gtk-convert")
```

`GTK_STOCK_CONVERT` has been deprecated since version 3.10 and should not be used in newly-written code.

The “Convert” item and icon.

GTK_STOCK_CONNECT

```
#define GTK_STOCK_CONNECT           ((GtkStock)"gtk-connect")
```

`GTK_STOCK_CONNECT` has been deprecated since version 3.10 and should not be used in newly-written code.

The “Connect” icon.

Since: 2.6

GTK_STOCK_COPY

```
#define GTK_STOCK_COPY              ((GtkStock)"gtk-copy")
```

`GTK_STOCK_COPY` has been deprecated since version 3.10 and should not be used in newly-written code.

Use the named icon "edit-copy" or the label "_Copy".

The “Copy” item and icon.

GTK_STOCK_CUT

```
#define GTK_STOCK_CUT               ((GtkStock)"gtk-cut")
```

`GTK_STOCK_CUT` has been deprecated since version 3.10 and should not be used in newly-written code.

Use the named icon "edit-cut" or the label "Cu_t".

The “Cut” item and icon.

GTK_STOCK_DELETE

```
#define GTK_STOCK_DELETE            ((GtkStock)"gtk-delete")
```

`GTK_STOCK_DELETE` has been deprecated since version 3.10 and should not be used in newly-written code.

Use the named icon "edit-delete" or the label "_Delete".

The “Delete” item and icon.

GTK_STOCK_DIALOG_AUTHENTICATION

```
#define GTK_STOCK_DIALOG_AUTHENTICATION ((GtkStock)"gtk-dialog-authentication")
```

`GTK_STOCK_DIALOG_AUTHENTICATION` has been deprecated since version 3.10 and should not be used in newly-written code.

Use named icon "dialog-password".

The “Authentication” item and icon.

Since: 2.4

GTK_STOCK_DIALOG_ERROR

```
#define GTK_STOCK_DIALOG_ERROR      ((GtkStock)"gtk-dialog-error")
```

`GTK_STOCK_DIALOG_ERROR` has been deprecated since version 3.10 and should not be used in newly-written code.

Use named icon "dialog-error".

The “Error” item and icon.

GTK_STOCK_DIALOG_INFO

```
#define GTK_STOCK_DIALOG_INFO       ((GtkStock)"gtk-dialog-info")
```

`GTK_STOCK_DIALOG_INFO` has been deprecated since version 3.10 and should not be used in newly-written code.

Use named icon "dialog-information".

The “Information” item and icon.

GTK_STOCK_DIALOG_QUESTION

```
#define GTK_STOCK_DIALOG_QUESTION   ((GtkStock)"gtk-dialog-question")
```

`GTK_STOCK_DIALOG_QUESTION` has been deprecated since version 3.10 and should not be used in newly-written code.

Use named icon "dialog-question".

The “Question” item and icon.

GTK_STOCK_DIALOG_WARNING

```
#define GTK_STOCK_DIALOG_WARNING    ((GtkStock)"gtk-dialog-warning")
```

`GTK_STOCK_DIALOG_WARNING` has been deprecated since version 3.10 and should not be used in newly-written code.

Use named icon "dialog-warning".

The “Warning” item and icon.

GTK_STOCK_DIRECTORY

```
#define GTK_STOCK_DIRECTORY         ((GtkStock)"gtk-directory")
```

`GTK_STOCK_DIRECTORY` has been deprecated since version 3.10 and should not be used in newly-written code.

Use named icon "folder".

The “Directory” icon.

Since: 2.6

GTK_STOCK_DISCARD

```
#define GTK_STOCK_DISCARD          ((GtkStock)"gtk-discard")
```

`GTK_STOCK_DISCARD` has been deprecated since version 3.10 and should not be used in newly-written code.

The “Discard” item.

Since: 2.12

GTK_STOCK_DISCONNECT

```
#define GTK_STOCK_DISCONNECT       ((GtkStock)"gtk-disconnect")
```

`GTK_STOCK_DISCONNECT` has been deprecated since version 3.10 and should not be used in newly-written code.

The “Disconnect” icon.

Since: 2.6

GTK_STOCK_DND

```
#define GTK_STOCK_DND              ((GtkStock)"gtk-dnd")
```

`GTK_STOCK_DND` has been deprecated since version 3.10 and should not be used in newly-written code.

The “Drag-And-Drop” icon.

GTK_STOCK_DND_MULTIPLE

```
#define GTK_STOCK_DND_MULTIPLE     ((GtkStock)"gtk-dnd-multiple")
```

`GTK_STOCK_DND_MULTIPLE` has been deprecated since version 3.10 and should not be used in newly-written code.

The “Drag-And-Drop multiple” icon.

GTK_STOCK_EDIT

```
#define GTK_STOCK_EDIT             ((GtkStock)"gtk-edit")
```

`GTK_STOCK_EDIT` has been deprecated since version 3.10 and should not be used in newly-written code.

The “Edit” item and icon.

Since: 2.6

GTK_STOCK_EXECUTE

```
#define GTK_STOCK_EXECUTE           ((GtkStock)"gtk-execute")
```

`GTK_STOCK_EXECUTE` has been deprecated since version 3.10 and should not be used in newly-written code.

Use named icon "system-run".

The “Execute” item and icon.

GTK_STOCK_FILE

```
#define GTK_STOCK_FILE              ((GtkStock)"gtk-file")
```

`GTK_STOCK_FILE` has been deprecated since version 3.10 and should not be used in newly-written code.

Use named icon "text-x-generic".

The “File” item and icon.

Since 3.0, this item has a label, before it only had an icon.

Since: 2.6

GTK_STOCK_FIND

```
#define GTK_STOCK_FIND              ((GtkStock)"gtk-find")
```

`GTK_STOCK_FIND` has been deprecated since version 3.10 and should not be used in newly-written code.

Use named icon "edit-find".

The “Find” item and icon.

GTK_STOCK_FIND_AND_REPLACE

```
#define GTK_STOCK_FIND_AND_REPLACE ((GtkStock)"gtk-find-and-replace")
```

`GTK_STOCK_FIND_AND_REPLACE` has been deprecated since version 3.10 and should not be used in newly-written code.

Use named icon "edit-find-replace".

The “Find and Replace” item and icon.

GTK_STOCK_FLOPPY

```
#define GTK_STOCK_FLOPPY            ((GtkStock)"gtk-floppy")
```

`GTK_STOCK_FLOPPY` has been deprecated since version 3.10 and should not be used in newly-written code.

The “Floppy” item and icon.

GTK_STOCK_FULLSCREEN

```
#define GTK_STOCK_FULLSCREEN       ((GtkStock)"gtk-fullscreen")
```

`GTK_STOCK_FULLSCREEN` has been deprecated since version 3.10 and should not be used in newly-written code.

Use named icon "view-fullscreen".

The “Fullscreen” item and icon.

Since: 2.8

GTK_STOCK_GOTO_BOTTOM

```
#define GTK_STOCK_GOTO_BOTTOM      ((GtkStock)"gtk-goto-bottom")
```

`GTK_STOCK_GOTO_BOTTOM` has been deprecated since version 3.10 and should not be used in newly-written code.

Use named icon "go-bottom".

The “Bottom” item and icon.

GTK_STOCK_GOTO_FIRST

```
#define GTK_STOCK_GOTO_FIRST       ((GtkStock)"gtk-goto-first")
```

`GTK_STOCK_GOTO_FIRST` has been deprecated since version 3.10 and should not be used in newly-written code.

Use named icon "go-first".

The “First” item and icon. The icon has an RTL variant.

GTK_STOCK_GOTO_LAST

```
#define GTK_STOCK_GOTO_LAST        ((GtkStock)"gtk-goto-last")
```

`GTK_STOCK_GOTO_LAST` has been deprecated since version 3.10 and should not be used in newly-written code.

Use named icon "go-last".

The “Last” item and icon. The icon has an RTL variant.

GTK_STOCK_GOTO_TOP

```
#define GTK_STOCK_GOTO_TOP         ((GtkStock)"gtk-goto-top")
```

`GTK_STOCK_GOTO_TOP` has been deprecated since version 3.10 and should not be used in newly-written code.

Use named icon "go-top".

The “Top” item and icon.

GTK_STOCK_GO_BACK

```
#define GTK_STOCK_GO_BACK           ((GtkStock)"gtk-go-back")
```

`GTK_STOCK_GO_BACK` has been deprecated since version 3.10 and should not be used in newly-written code.

Use named icon "go-previous".

The “Back” item and icon. The icon has an RTL variant.

GTK_STOCK_GO_DOWN

```
#define GTK_STOCK_GO_DOWN           ((GtkStock)"gtk-go-down")
```

`GTK_STOCK_GO_DOWN` has been deprecated since version 3.10 and should not be used in newly-written code.

Use named icon "go-down".

The “Down” item and icon.

GTK_STOCK_GO_FORWARD

```
#define GTK_STOCK_GO_FORWARD        ((GtkStock)"gtk-go-forward")
```

`GTK_STOCK_GO_FORWARD` has been deprecated since version 3.10 and should not be used in newly-written code.

Use named icon "go-next".

The “Forward” item and icon. The icon has an RTL variant.

GTK_STOCK_GO_UP

```
#define GTK_STOCK_GO_UP             ((GtkStock)"gtk-go-up")
```

`GTK_STOCK_GO_UP` has been deprecated since version 3.10 and should not be used in newly-written code.

Use named icon "go-up".

The “Up” item and icon.

GTK_STOCK_HARDDISK

```
#define GTK_STOCK_HARDDISK          ((GtkStock)"gtk-harddisk")
```

`GTK_STOCK_HARDDISK` has been deprecated since version 3.10 and should not be used in newly-written code.

Use named icon "drive-harddisk".

The “Harddisk” item and icon.

Since: 2.4

GTK_STOCK_HELP

```
#define GTK_STOCK_HELP              ((GtkStock)"gtk-help")
```

`GTK_STOCK_HELP` has been deprecated since version 3.10 and should not be used in newly-written code.

Use named icon "help-browser".

The “Help” item and icon.

GTK_STOCK_HOME

```
#define GTK_STOCK_HOME              ((GtkStock)"gtk-home")
```

`GTK_STOCK_HOME` has been deprecated since version 3.10 and should not be used in newly-written code.

Use named icon "go-home".

The “Home” item and icon.

GTK_STOCK_INDENT

```
#define GTK_STOCK_INDENT            ((GtkStock)"gtk-indent")
```

`GTK_STOCK_INDENT` has been deprecated since version 3.10 and should not be used in newly-written code.

Use named icon "format-indent-more".

The “Indent” item and icon. The icon has an RTL variant.

Since: 2.4

GTK_STOCK_INDEX

```
#define GTK_STOCK_INDEX             ((GtkStock)"gtk-index")
```

`GTK_STOCK_INDEX` has been deprecated since version 3.10 and should not be used in newly-written code.

The “Index” item and icon.

GTK_STOCK_INFO

```
#define GTK_STOCK_INFO              ((GtkStock)"gtk-info")
```

`GTK_STOCK_INFO` has been deprecated since version 3.10 and should not be used in newly-written code.

Use named icon "dialog-information".

The “Info” item and icon.

Since: 2.8

GTK_STOCK_ITALIC

```
#define GTK_STOCK_ITALIC            ((GtkStock)"gtk-italic")
```

`GTK_STOCK_ITALIC` has been deprecated since version 3.10 and should not be used in newly-written code.

Use named icon "format-text-italic".

The “Italic” item and icon.

GTK_STOCK_JUMP_TO

```
#define GTK_STOCK_JUMP_TO           ((GtkStock)"gtk-jump-to")
```

`GTK_STOCK_JUMP_TO` has been deprecated since version 3.10 and should not be used in newly-written code.

Use named icon "go-jump".

The “Jump to” item and icon. The icon has an RTL variant.

GTK_STOCK_JUSTIFY_CENTER

```
#define GTK_STOCK_JUSTIFY_CENTER    ((GtkStock)"gtk-justify-center")
```

`GTK_STOCK_JUSTIFY_CENTER` has been deprecated since version 3.10 and should not be used in newly-written code.

Use named icon "format-justify-center".

The “Center” item and icon.

GTK_STOCK_JUSTIFY_FILL

```
#define GTK_STOCK_JUSTIFY_FILL      ((GtkStock)"gtk-justify-fill")
```

`GTK_STOCK_JUSTIFY_FILL` has been deprecated since version 3.10 and should not be used in newly-written code.

Use named icon "format-justify-fill".

The “Fill” item and icon.

GTK_STOCK_JUSTIFY_LEFT

```
#define GTK_STOCK_JUSTIFY_LEFT      ((GtkStock)"gtk-justify-left")
```

`GTK_STOCK_JUSTIFY_LEFT` has been deprecated since version 3.10 and should not be used in newly-written code.

Use named icon "format-justify-left".

The “Left” item and icon.

GTK_STOCK_JUSTIFY_RIGHT

```
#define GTK_STOCK_JUSTIFY_RIGHT    ((GtkStock)"gtk-justify-right")
```

`GTK_STOCK_JUSTIFY_RIGHT` has been deprecated since version 3.10 and should not be used in newly-written code.

Use named icon "format-justify-right".

The “Right” item and icon.

GTK_STOCK_LEAVE_FULLSCREEN

```
#define GTK_STOCK_LEAVE_FULLSCREEN ((GtkStock)"gtk-leave-fullscreen")
```

`GTK_STOCK_LEAVE_FULLSCREEN` has been deprecated since version 3.10 and should not be used in newly-written code.

Use named icon "view-restore".

The “Leave Fullscreen” item and icon.

Since: 2.8

GTK_STOCK_MEDIA_FORWARD

```
#define GTK_STOCK_MEDIA_FORWARD    ((GtkStock)"gtk-media-forward")
```

`GTK_STOCK_MEDIA_FORWARD` has been deprecated since version 3.10 and should not be used in newly-written code.

Use named icon "media-seek-forward" or the label "_Forward".

The “Media Forward” item and icon. The icon has an RTL variant.

Since: 2.6

GTK_STOCK_MEDIA_NEXT

```
#define GTK_STOCK_MEDIA_NEXT       ((GtkStock)"gtk-media-next")
```

`GTK_STOCK_MEDIA_NEXT` has been deprecated since version 3.10 and should not be used in newly-written code.

Use named icon "media-skip-forward" or the label "_Next".

The “Media Next” item and icon. The icon has an RTL variant.

Since: 2.6

GTK_STOCK_MEDIA_PAUSE

```
#define GTK_STOCK_MEDIA_PAUSE      ((GtkStock)"gtk-media-pause")
```

`GTK_STOCK_MEDIA_PAUSE` has been deprecated since version 3.10 and should not be used in newly-written code.

Use named icon "media-playback-pause" or the label "P_ause".

The “Media Pause” item and icon.

Since: 2.6

GTK_STOCK_MEDIA_PLAY

```
#define GTK_STOCK_MEDIA_PLAY       ((GtkStock)"gtk-media-play")
```

`GTK_STOCK_MEDIA_PLAY` has been deprecated since version 3.10 and should not be used in newly-written code.

Use named icon "media-playback-start" or the label "_Play".

The “Media Play” item and icon. The icon has an RTL variant.

Since: 2.6

GTK_STOCK_MEDIA_PREVIOUS

```
#define GTK_STOCK_MEDIA_PREVIOUS   ((GtkStock)"gtk-media-previous")
```

`GTK_STOCK_MEDIA_PREVIOUS` has been deprecated since version 3.10 and should not be used in newly-written code.

Use named icon "media-skip-backward" or the label "Pre_vious".

The “Media Previous” item and icon. The icon has an RTL variant.

Since: 2.6

GTK_STOCK_MEDIA_RECORD

```
#define GTK_STOCK_MEDIA_RECORD     ((GtkStock)"gtk-media-record")
```

`GTK_STOCK_MEDIA_RECORD` has been deprecated since version 3.10 and should not be used in newly-written code.

Use named icon "media-record" or the label "_Record".

The “Media Record” item and icon.

Since: 2.6

GTK_STOCK_MEDIA_REWIND

```
#define GTK_STOCK_MEDIA_REWIND     ((GtkStock)"gtk-media-rewind")
```

`GTK_STOCK_MEDIA_REWIND` has been deprecated since version 3.10 and should not be used in newly-written code.

Use named icon "media-seek-backward" or the label "R_ewind".

The “Media Rewind” item and icon. The icon has an RTL variant.

Since: 2.6

GTK_STOCK_MEDIA_STOP

```
#define GTK_STOCK_MEDIA_STOP       ((GtkStock)"gtk-media-stop")
```

`GTK_STOCK_MEDIA_STOP` has been deprecated since version 3.10 and should not be used in newly-written code.

Use named icon "media-playback-stop" or the label "_Stop".

The “Media Stop” item and icon.

Since: 2.6

GTK_STOCK_MISSING_IMAGE

```
#define GTK_STOCK_MISSING_IMAGE    ((GtkStock)"gtk-missing-image")
```

`GTK_STOCK_MISSING_IMAGE` has been deprecated since version 3.10 and should not be used in newly-written code.

Use named icon "image-missing".

The “Missing image” icon.

GTK_STOCK_NETWORK

```
#define GTK_STOCK_NETWORK          ((GtkStock)"gtk-network")
```

`GTK_STOCK_NETWORK` has been deprecated since version 3.10 and should not be used in newly-written code.

Use named icon "network-workgroup".

The “Network” item and icon.

Since: 2.4

GTK_STOCK_NEW

```
#define GTK_STOCK_NEW              ((GtkStock)"gtk-new")
```

`GTK_STOCK_NEW` has been deprecated since version 3.10 and should not be used in newly-written code.

Use named icon "document-new" or the label "_New".

The “New” item and icon.

GTK_STOCK_NO

```
#define GTK_STOCK_NO                ((GtkStock)"gtk-no")
```

`GTK_STOCK_NO` has been deprecated since version 3.10 and should not be used in newly-written code.

The “No” item and icon.

GTK_STOCK_OK

```
#define GTK_STOCK_OK                ((GtkStock)"gtk-ok")
```

`GTK_STOCK_OK` has been deprecated since version 3.10 and should not be used in newly-written code.

Do not use an icon. Use label "_OK".

The “OK” item and icon.

GTK_STOCK_OPEN

```
#define GTK_STOCK_OPEN              ((GtkStock)"gtk-open")
```

`GTK_STOCK_OPEN` has been deprecated since version 3.10 and should not be used in newly-written code.

Use named icon "document-open" or the label "_Open".

The “Open” item and icon.

GTK_STOCK_ORIENTATION_LANDSCAPE

```
#define GTK_STOCK_ORIENTATION_LANDSCAPE ((GtkStock)"gtk-orientation-landscape")
```

`GTK_STOCK_ORIENTATION_LANDSCAPE` has been deprecated since version 3.10 and should not be used in newly-written code.

The “Landscape Orientation” item and icon.

Since: 2.10

GTK_STOCK_ORIENTATION_PORTRAIT

```
#define GTK_STOCK_ORIENTATION_PORTRAIT ((GtkStock)"gtk-orientation-portrait")
```

`GTK_STOCK_ORIENTATION_PORTRAIT` has been deprecated since version 3.10 and should not be used in newly-written code.

The “Portrait Orientation” item and icon.

Since: 2.10

GTK_STOCK_ORIENTATION_REVERSE_LANDSCAPE

```
#define GTK_STOCK_ORIENTATION_REVERSE_LANDSCAPE ((GtkStock)"gtk-orientation-reverse-landscape")
```

`GTK_STOCK_ORIENTATION_REVERSE_LANDSCAPE` has been deprecated since version 3.10 and should not be used in newly-written code.

The “Reverse Landscape Orientation” item and icon.

Since: 2.10

GTK_STOCK_ORIENTATION_REVERSE_PORTRAIT

```
#define GTK_STOCK_ORIENTATION_REVERSE_PORTRAIT ((GtkStock)"gtk-orientation-reverse-portrait")
```

`GTK_STOCK_ORIENTATION_REVERSE_PORTRAIT` has been deprecated since version 3.10 and should not be used in newly-written code.

The “Reverse Portrait Orientation” item and icon.

Since: 2.10

GTK_STOCK_PAGE_SETUP

```
#define GTK_STOCK_PAGE_SETUP       ((GtkStock)"gtk-page-setup")
```

`GTK_STOCK_PAGE_SETUP` has been deprecated since version 3.10 and should not be used in newly-written code.

Use named icon "document-page-setup" or the label "Page Set_up".

The “Page Setup” item and icon.

Since: 2.14

GTK_STOCK_PASTE

```
#define GTK_STOCK_PASTE            ((GtkStock)"gtk-paste")
```

`GTK_STOCK_PASTE` has been deprecated since version 3.10 and should not be used in newly-written code.

Use named icon "edit-paste" or the label "_Paste".

The “Paste” item and icon.

GTK_STOCK_PREFERENCES

```
#define GTK_STOCK_PREFERENCES      ((GtkStock)"gtk-preferences")
```

`GTK_STOCK_PREFERENCES` has been deprecated since version 3.10 and should not be used in newly-written code.

Use named icon "preferences-system" or the label "_Preferences".

The “Preferences” item and icon.

GTK_STOCK_PRINT

```
#define GTK_STOCK_PRINT            ((GtkStock)"gtk-print")
```

`GTK_STOCK_PRINT` has been deprecated since version 3.10 and should not be used in newly-written code.

Use named icon "document-print" or the label "_Print".

The “Print” item and icon.

GTK_STOCK_PRINT_ERROR

```
#define GTK_STOCK_PRINT_ERROR      ((GtkStock)"gtk-print-error")
```

`GTK_STOCK_PRINT_ERROR` has been deprecated since version 3.10 and should not be used in newly-written code.

Use named icon "printer-error".

The “Print Error” icon.

Since: 2.14

GTK_STOCK_PRINT_PAUSED

```
#define GTK_STOCK_PRINT_PAUSED     ((GtkStock)"gtk-print-paused")
```

`GTK_STOCK_PRINT_PAUSED` has been deprecated since version 3.10 and should not be used in newly-written code.

The “Print Paused” icon.

Since: 2.14

GTK_STOCK_PRINT_PREVIEW

```
#define GTK_STOCK_PRINT_PREVIEW    ((GtkStock)"gtk-print-preview")
```

`GTK_STOCK_PRINT_PREVIEW` has been deprecated since version 3.10 and should not be used in newly-written code.

Use label "Pre_view".

The “Print Preview” item and icon.

GTK_STOCK_PRINT_REPORT

```
#define GTK_STOCK_PRINT_REPORT     ((GtkStock)"gtk-print-report")
```

`GTK_STOCK_PRINT_REPORT` has been deprecated since version 3.10 and should not be used in newly-written code.

The “Print Report” icon.

Since: 2.14

GTK_STOCK_PRINT_WARNING

```
#define GTK_STOCK_PRINT_WARNING    ((GtkStock)"gtk-print-warning")
```

`GTK_STOCK_PRINT_WARNING` has been deprecated since version 3.10 and should not be used in newly-written code.

The “Print Warning” icon.

Since: 2.14

GTK_STOCK_PROPERTIES

```
#define GTK_STOCK_PROPERTIES       ((GtkStock)"gtk-properties")
```

`GTK_STOCK_PROPERTIES` has been deprecated since version 3.10 and should not be used in newly-written code.

Use named icon "document-properties" or the label "_Properties".

The “Properties” item and icon.

GTK_STOCK_QUIT

```
#define GTK_STOCK_QUIT             ((GtkStock)"gtk-quit")
```

`GTK_STOCK_QUIT` has been deprecated since version 3.10 and should not be used in newly-written code.

Use named icon "application-exit" or the label "_Quit".

The “Quit” item and icon.

GTK_STOCK_REDO

```
#define GTK_STOCK_REDO             ((GtkStock)"gtk-redo")
```

`GTK_STOCK_REDO` has been deprecated since version 3.10 and should not be used in newly-written code.

Use named icon "edit-redo" or the label "_Redo".

The “Redo” item and icon. The icon has an RTL variant.

GTK_STOCK_REFRESH

```
#define GTK_STOCK_REFRESH           ((GtkStock)"gtk-refresh")
```

`GTK_STOCK_REFRESH` has been deprecated since version 3.10 and should not be used in newly-written code.

Use named icon "view-refresh" or the label "_Refresh".

The “Refresh” item and icon.

GTK_STOCK_REMOVE

```
#define GTK_STOCK_REMOVE            ((GtkStock)"gtk-remove")
```

`GTK_STOCK_REMOVE` has been deprecated since version 3.10 and should not be used in newly-written code.

Use named icon "list-remove" or the label "_Remove".

The “Remove” item and icon.

GTK_STOCK_REVERT_TO_SAVED

```
#define GTK_STOCK_REVERT_TO_SAVED  ((GtkStock)"gtk-revert-to-saved")
```

`GTK_STOCK_REVERT_TO_SAVED` has been deprecated since version 3.10 and should not be used in newly-written code.

Use named icon "document-revert" or the label "_Revert".

The “Revert” item and icon. The icon has an RTL variant.

GTK_STOCK_SAVE

```
#define GTK_STOCK_SAVE              ((GtkStock)"gtk-save")
```

`GTK_STOCK_SAVE` has been deprecated since version 3.10 and should not be used in newly-written code.

Use named icon "document-save" or the label "_Save".

The “Save” item and icon.

GTK_STOCK_SAVE_AS

```
#define GTK_STOCK_SAVE_AS           ((GtkStock)"gtk-save-as")
```

`GTK_STOCK_SAVE_AS` has been deprecated since version 3.10 and should not be used in newly-written code.

Use named icon "document-save-as" or the label "Save _As".

The “Save As” item and icon.

GTK_STOCK_SELECT_ALL

```
#define GTK_STOCK_SELECT_ALL       ((GtkStock)"gtk-select-all")
```

`GTK_STOCK_SELECT_ALL` has been deprecated since version 3.10 and should not be used in newly-written code.

Use named icon "edit-select-all" or the label "Select _All".

The “Select All” item and icon.

Since: 2.10

GTK_STOCK_SELECT_COLOR

```
#define GTK_STOCK_SELECT_COLOR     ((GtkStock)"gtk-select-color")
```

`GTK_STOCK_SELECT_COLOR` has been deprecated since version 3.10 and should not be used in newly-written code.

The “Color” item and icon.

GTK_STOCK_SELECT_FONT

```
#define GTK_STOCK_SELECT_FONT      ((GtkStock)"gtk-select-font")
```

`GTK_STOCK_SELECT_FONT` has been deprecated since version 3.10 and should not be used in newly-written code.

The “Font” item and icon.

GTK_STOCK_SORT_ASCENDING

```
#define GTK_STOCK_SORT_ASCENDING   ((GtkStock)"gtk-sort-ascending")
```

`GTK_STOCK_SORT_ASCENDING` has been deprecated since version 3.10 and should not be used in newly-written code.

Use named icon "view-sort-ascending".

The “Ascending” item and icon.

GTK_STOCK_SORT_DESCENDING

```
#define GTK_STOCK_SORT_DESCENDING  ((GtkStock)"gtk-sort-descending")
```

`GTK_STOCK_SORT_DESCENDING` has been deprecated since version 3.10 and should not be used in newly-written code.

Use named icon "view-sort-descending".

The “Descending” item and icon.

GTK_STOCK_SPELL_CHECK

```
#define GTK_STOCK_SPELL_CHECK      ((GtkStock)"gtk-spell-check")
```

`GTK_STOCK_SPELL_CHECK` has been deprecated since version 3.10 and should not be used in newly-written code.

Use named icon "tools-check-spelling".

The “Spell Check” item and icon.

GTK_STOCK_STOP

```
#define GTK_STOCK_STOP             ((GtkStock)"gtk-stop")
```

`GTK_STOCK_STOP` has been deprecated since version 3.10 and should not be used in newly-written code.

Use named icon "process-stop" or the label "_Stop".

The “Stop” item and icon.

GTK_STOCK_STRIKETHROUGH

```
#define GTK_STOCK_STRIKETHROUGH    ((GtkStock)"gtk-strikethrough")
```

`GTK_STOCK_STRIKETHROUGH` has been deprecated since version 3.10 and should not be used in newly-written code.

Use named icon "format-text-strikethrough" or the label "_Strikethrough".

The “Strikethrough” item and icon.

GTK_STOCK_UNDELETE

```
#define GTK_STOCK_UNDELETE         ((GtkStock)"gtk-undelete")
```

`GTK_STOCK_UNDELETE` has been deprecated since version 3.10 and should not be used in newly-written code.

The “Undelete” item and icon. The icon has an RTL variant.

GTK_STOCK_UNDERLINE

```
#define GTK_STOCK_UNDERLINE        ((GtkStock)"gtk-underline")
```

`GTK_STOCK_UNDERLINE` has been deprecated since version 3.10 and should not be used in newly-written code.

Use named icon "format-text-underline" or the label "_Underline".

The “Underline” item and icon.

GTK_STOCK_UNDO

```
#define GTK_STOCK_UNDO              ((GtkStock)"gtk-undo")
```

`GTK_STOCK_UNDO` has been deprecated since version 3.10 and should not be used in newly-written code.

Use named icon "edit-undo" or the label "_Undo".

The “Undo” item and icon. The icon has an RTL variant.

GTK_STOCK_UNINDENT

```
#define GTK_STOCK_UNINDENT          ((GtkStock)"gtk-unindent")
```

`GTK_STOCK_UNINDENT` has been deprecated since version 3.10 and should not be used in newly-written code.

Use named icon "format-indent-less".

The “Unindent” item and icon. The icon has an RTL variant.

Since: 2.4

GTK_STOCK_YES

```
#define GTK_STOCK_YES               ((GtkStock)"gtk-yes")
```

`GTK_STOCK_YES` has been deprecated since version 3.10 and should not be used in newly-written code.

The “Yes” item and icon.

GTK_STOCK_ZOOM_100

```
#define GTK_STOCK_ZOOM_100          ((GtkStock)"gtk-zoom-100")
```

`GTK_STOCK_ZOOM_100` has been deprecated since version 3.10 and should not be used in newly-written code.

Use named icon "zoom-original" or the label "_Normal Size".

The “Zoom 100%” item and icon.

GTK_STOCK_ZOOM_FIT

```
#define GTK_STOCK_ZOOM_FIT          ((GtkStock)"gtk-zoom-fit")
```

`GTK_STOCK_ZOOM_FIT` has been deprecated since version 3.10 and should not be used in newly-written code.

Use named icon "zoom-fit-best" or the label "Best _Fit".

The “Zoom to Fit” item and icon.

GTK_STOCK_ZOOM_IN

```
#define GTK_STOCK_ZOOM_IN          ((GtkStock)"gtk-zoom-in")
```

`GTK_STOCK_ZOOM_IN` has been deprecated since version 3.10 and should not be used in newly-written code.

Use named icon "zoom-in" or the label "Zoom _In".

The “Zoom In” item and icon.

GTK_STOCK_ZOOM_OUT

```
#define GTK_STOCK_ZOOM_OUT         ((GtkStock)"gtk-zoom-out")
```

`GTK_STOCK_ZOOM_OUT` has been deprecated since version 3.10 and should not be used in newly-written code.

Use named icon "zoom-out" or the label "Zoom _Out".

The “Zoom Out” item and icon.

Themeable Stock Images

Themeable Stock Images — Manipulating stock icons

Functions

GtkIconSource *	gtk_icon_source_copy ()
void	gtk_icon_source_free ()
void	gtk_icon_factory_add ()
void	gtk_icon_factory_add_default ()
GtkIconSet *	gtk_icon_factory_lookup ()
GtkIconSet *	gtk_icon_factory_lookup_default ()
GtkIconFactory *	gtk_icon_factory_new ()
void	gtk_icon_factory_remove_default ()
void	gtk_icon_set_add_source ()
GtkIconSet *	gtk_icon_set_copy ()
GtkIconSet *	gtk_icon_set_new ()
GtkIconSet *	gtk_icon_set_new_from_pixbuf ()
GtkIconSet *	gtk_icon_set_ref ()
GdkPixbuf *	gtk_icon_set_render_icon ()
GdkPixbuf *	gtk_icon_set_render_icon_pixbuf ()
cairo_surface_t *	gtk_icon_set_render_icon_surface ()
void	gtk_icon_set_unref ()
gboolean	gtk_icon_size_lookup ()
gboolean	gtk_icon_size_lookup_for_settings ()
GtkIconSize	gtk_icon_size_register ()
void	gtk_icon_size_register_alias ()
GtkIconSize	gtk_icon_size_from_name ()
const gchar *	gtk_icon_size_get_name ()
void	gtk_icon_set_get_sizes ()
GtkTextDirection	gtk_icon_source_get_direction ()

gboolean	gtk_icon_source_get_direction_wildcarded ()
const gchar *	gtk_icon_source_get_filename ()
GdkPixbuf *	gtk_icon_source_get_pixbuf ()
const gchar *	gtk_icon_source_get_icon_name ()
GtkIconSize	gtk_icon_source_get_size ()
gboolean	gtk_icon_source_get_size_wildcarded ()
GtkStateType	gtk_icon_source_get_state ()
gboolean	gtk_icon_source_get_state_wildcarded ()
GtkIconSource *	gtk_icon_source_new ()
void	gtk_icon_source_set_direction ()
void	gtk_icon_source_set_direction_wildcarded ()
void	gtk_icon_source_set_filename ()
void	gtk_icon_source_set_pixbuf ()
void	gtk_icon_source_set_icon_name ()
void	gtk_icon_source_set_size ()
void	gtk_icon_source_set_size_wildcarded ()
void	gtk_icon_source_set_state ()
void	gtk_icon_source_set_state_wildcarded ()

Types and Values

	GtkIconSource
struct	GtkIconFactory
struct	GtkIconFactoryClass
	GtkIconSet
enum	GtkIconSize

Object Hierarchy

```
GBoxed
└── GtkIconSet
GObject
└── GtkIconFactory
```

Implemented Interfaces

GtkIconFactory implements GtkBuildable.

Includes

```
#include <gtk/gtk.h>
```

Description

An icon factory manages a collection of GtkIconSet; a GtkIconSet manages a set of variants of a particular icon (i.e. a GtkIconSet contains variants for different sizes and widget states). Icons in an icon factory are named by a stock ID, which is a simple string identifying the icon. Each GtkStyle has a list of GtkIconFactory derived from the current theme; those icon factories are consulted first when searching for an icon. If the theme doesn’t

set a particular icon, GTK+ looks for the icon in a list of default icon factories, maintained by `gtk_icon_factory_add_default()` and `gtk_icon_factory_remove_default()`. Applications with icons should add a default icon factory with their icons, which will allow themes to override the icons for the application.

To display an icon, always use `gtk_style_lookup_icon_set()` on the widget that will display the icon, or the convenience function `gtk_widget_render_icon()`. These functions take the theme into account when looking up the icon to use for a given stock ID.

GtkIconFactory as GtkBuildable

GtkIconFactory supports a custom <sources> element, which can contain multiple <source> elements. The following attributes are allowed:

- stock-id

  The stock id of the source, a string. This attribute is mandatory

- filename

  The filename of the source, a string. This attribute is optional

- icon-name

  The icon name for the source, a string. This attribute is optional.

- size

  Size of the icon, a GtkIconSize enum value. This attribute is optional.

- direction

  Direction of the source, a GtkTextDirection enum value. This attribute is optional.

- state

  State of the source, a GtkStateType enum value. This attribute is optional.

A GtkIconFactory UI definition fragment.

```
<object class="GtkIconFactory"
id="iconfactory1">
  <sources>
    <source stock-id="apple-red"
filename="apple-red.png"/>
  </sources>
</object>
<object class="GtkWindow" id="window1">
  <child>
    <object class="GtkButton"
id="apple_button">
      <property
name="label">apple-red</property>
      <property
name="use-stock">True</property>
    </object>
  </child>
</object>
```

Functions

gtk_icon_source_copy ()

```
GtkIconSource *
gtk_icon_source_copy (const GtkIconSource *source);
```

`gtk_icon_source_copy` has been deprecated since version 3.10 and should not be used in newly-written code.

Use GtkIconTheme instead.

Creates a copy of `source` ; mostly useful for language bindings.

Parameters

source	a GtkIconSource

Returns

a new GtkIconSource

---

gtk_icon_source_free ()

```
void
gtk_icon_source_free (GtkIconSource *source);
```

`gtk_icon_source_free` has been deprecated since version 3.10 and should not be used in newly-written code.

Use GtkIconTheme instead.

Frees a dynamically-allocated icon source, along with its filename, size, and pixbuf fields if those are not `NULL`.

Parameters

source	a GtkIconSource

---

gtk_icon_factory_add ()

```
void
gtk_icon_factory_add (GtkIconFactory *factory,
                      const gchar *stock_id,
                      GtkIconSet *icon_set);
```

`gtk_icon_factory_add` has been deprecated since version 3.10 and should not be used in newly-written code.

Use GtkIconTheme instead.

Adds the given `icon_set` to the icon factory, under the name `stock_id` . `stock_id` should be namespaced for your application, e.g. “myapp-whatever-icon”. Normally applications create a GtkIconFactory, then add it to the

list of default factories with gtk_icon_factory_add_default(). Then they pass the stock_id to widgets such as GtkImage to display the icon. Themes can provide an icon with the same name (such as "myapp-whatever-icon") to override your application's default icons. If an icon already existed in factory for stock_id , it is unreferenced and replaced with the new icon_set .

Parameters

factory	a GtkIconFactory
stock_id	icon name
icon_set	icon set

---

gtk_icon_factory_add_default ()

```
void
gtk_icon_factory_add_default (GtkIconFactory *factory);
```

gtk_icon_factory_add_default has been deprecated since version 3.10 and should not be used in newly-written code.

Use GtkIconTheme instead.

Adds an icon factory to the list of icon factories searched by gtk_style_lookup_icon_set(). This means that, for example, gtk_image_new_from_stock() will be able to find icons in factory . There will normally be an icon factory added for each library or application that comes with icons. The default icon factories can be overridden by themes.

Parameters

factory	a GtkIconFactory

---

gtk_icon_factory_lookup ()

```
GtkIconSet *
gtk_icon_factory_lookup (GtkIconFactory *factory,
                         const gchar *stock_id);
```

gtk_icon_factory_lookup has been deprecated since version 3.10 and should not be used in newly-written code.

Use GtkIconTheme instead.

Looks up stock_id in the icon factory, returning an icon set if found, otherwise NULL. For display to the user, you should use gtk_style_lookup_icon_set() on the GtkStyle for the widget that will display the icon, instead of using this function directly, so that themes are taken into account.

Parameters

factory	a GtkIconFactory
stock_id	an icon name

Returns

icon set of `stock_id`.

[transfer none]

---

gtk_icon_factory_lookup_default ()

```
GtkIconSet *
gtk_icon_factory_lookup_default (const gchar *stock_id);
```

`gtk_icon_factory_lookup_default` has been deprecated since version 3.10 and should not be used in newly-written code.

Use GtkIconTheme instead.

Looks for an icon in the list of default icon factories. For display to the user, you should use `gtk_style_lookup_icon_set()` on the GtkStyle for the widget that will display the icon, instead of using this function directly, so that themes are taken into account.

Parameters

stock_id	an icon name

Returns

a GtkIconSet, or `NULL`.

[transfer none]

---

gtk_icon_factory_new ()

```
GtkIconFactory *
gtk_icon_factory_new (void);
```

`gtk_icon_factory_new` has been deprecated since version 3.10 and should not be used in newly-written code.

Use GtkIconTheme instead.

Creates a new GtkIconFactory. An icon factory manages a collection of GtkIconSets; a GtkIconSet manages a set of variants of a particular icon (i.e. a GtkIconSet contains variants for different sizes and widget states). Icons in an icon factory are named by a stock ID, which is a simple string identifying the icon. Each GtkStyle has a list of GtkIconFactorys derived from the current theme; those icon factories are consulted first when searching for an icon. If the theme doesn't set a particular icon, GTK+ looks for the icon in a list of default icon factories, maintained by `gtk_icon_factory_add_default()` and `gtk_icon_factory_remove_default()`. Applications with icons should add a default icon factory with their icons, which will allow themes to override the icons for the application.

Returns

a new GtkIconFactory

---

gtk_icon_factory_remove_default ()

```
void
gtk_icon_factory_remove_default (GtkIconFactory *factory);
```

`gtk_icon_factory_remove_default` has been deprecated since version 3.10 and should not be used in newly-written code.

Use GtkIconTheme instead.

Removes an icon factory from the list of default icon factories. Not normally used; you might use it for a library that can be unloaded or shut down.

Parameters

factory	a GtkIconFactory previously added with `gtk_icon_factory_add_default()`

---

gtk_icon_set_add_source ()

```
void
gtk_icon_set_add_source (GtkIconSet *icon_set,
                         const GtkIconSource *source);
```

`gtk_icon_set_add_source` has been deprecated since version 3.10 and should not be used in newly-written code.

Use GtkIconTheme instead.

Icon sets have a list of GtkIconSource, which they use as base icons for rendering icons in different states and sizes. Icons are scaled, made to look insensitive, etc. in `gtk_icon_set_render_icon()`, but GtkIconSet needs base images to work with. The base images and when to use them are described by a GtkIconSource.

This function copies `source` , so you can reuse the same source immediately without affecting the icon set.

An example of when you'd use this function: a web browser's "Back to Previous Page" icon might point in a different direction in Hebrew and in English; it might look different when insensitive; and it might change size depending on toolbar mode (small/large icons). So a single icon set would contain all those variants of the icon, and you might add a separate source for each one.

You should nearly always add a "default" icon source with all fields wildcarded, which will be used as a fallback if no more specific source matches. GtkIconSet always prefers more specific icon sources to more generic icon sources. The order in which you add the sources to the icon set does not matter.

gtk_icon_set_new_from_pixbuf() creates a new icon set with a default icon source based on the given pixbuf.

Parameters

icon_set	a GtkIconSet
source	a GtkIconSource

---

gtk_icon_set_copy ()

```
GtkIconSet *
gtk_icon_set_copy (GtkIconSet *icon_set);
```

`gtk_icon_set_copy` has been deprecated since version 3.10 and should not be used in newly-written code.

Use GtkIconTheme instead.

Copies `icon_set` by value.

Parameters

icon_set	a GtkIconSet

Returns

a new GtkIconSet identical to the first.

---

gtk_icon_set_new ()

```
GtkIconSet *
gtk_icon_set_new (void);
```

`gtk_icon_set_new` has been deprecated since version 3.10 and should not be used in newly-written code.

Use GtkIconTheme instead.

Creates a new GtkIconSet. A GtkIconSet represents a single icon in various sizes and widget states. It can provide a GdkPixbuf for a given size and state on request, and automatically caches some of the rendered GdkPixbuf objects.

Normally you would use `gtk_widget_render_icon_pixbuf()` instead of using GtkIconSet directly. The one case where you'd use GtkIconSet is to create application-specific icon sets to place in a GtkIconFactory.

Returns

a new GtkIconSet

---

gtk_icon_set_new_from_pixbuf ()

```
GtkIconSet *
gtk_icon_set_new_from_pixbuf (GdkPixbuf *pixbuf);
```

`gtk_icon_set_new_from_pixbuf` has been deprecated since version 3.10 and should not be used in newly-

written code.

Use GtkIconTheme instead.

Creates a new GtkIconSet with `pixbuf` as the default/fallback source image. If you don’t add any additional GtkIconSource to the icon set, all variants of the icon will be created from `pixbuf` , using scaling, pixelation, etc. as required to adjust the icon size or make the icon look insensitive/prelighted.

Parameters

pixbuf	a GdkPixbuf

Returns

a new GtkIconSet

---

gtk_icon_set_ref ()

```
GtkIconSet *
gtk_icon_set_ref (GtkIconSet *icon_set);
```

`gtk_icon_set_ref` has been deprecated since version 3.10 and should not be used in newly-written code.

Use GtkIconTheme instead.

Increments the reference count on `icon_set` .

Parameters

icon_set	a GtkIconSet.

Returns

`icon_set` .

---

gtk_icon_set_render_icon ()

```
GdkPixbuf *
gtk_icon_set_render_icon (GtkIconSet *icon_set,
                          GtkStyle *style,
                          GtkTextDirection direction,
                          GtkStateType state,
                          GtkIconSize size,
                          GtkWidget *widget,
                          const gchar *detail);
```

`gtk_icon_set_render_icon` has been deprecated since version 3.0 and should not be used in newly-written code.

Use `gtk_icon_set_render_icon_pixbuf()` instead

Renders an icon using `gtk_style_render_icon()`. In most cases, `gtk_widget_render_icon()` is better, since it automatically provides most of the arguments from the current widget settings. This function never returns `NULL`; if the icon can’t be rendered (perhaps because an image file fails to load), a default "missing image" icon will be returned instead.

Parameters

icon_set	a GtkIconSet	
style	a GtkStyle associated with `widget` , or `NULL`.	[allow-none]
direction	text direction	
state	widget state	
size	icon size (GtkIconSize). A size of `(GtkIconSize)-1` means render at the size of the source and don’t scale.	[type int]
widget	widget that will display the icon, or `NULL`. The only use that is typically made of this is to determine the appropriate GdkScreen.	[allow-none]
detail	detail to pass to the theme engine, or `NULL`. Note that passing a detail of anything but `NULL` will disable caching.	[allow-none]

Returns

a GdkPixbuf to be displayed.

[transfer full]

---

gtk_icon_set_render_icon_pixbuf ()

```
GdkPixbuf *
gtk_icon_set_render_icon_pixbuf (GtkIconSet *icon_set,
                                 GtkStyleContext *context,
                                 GtkIconSize size);
```

`gtk_icon_set_render_icon_pixbuf` has been deprecated since version 3.10 and should not be used in newly-written code.

Use GtkIconTheme instead.

Renders an icon using `gtk_render_icon_pixbuf()`. In most cases, `gtk_widget_render_icon_pixbuf()` is better, since it automatically provides most of the arguments from the current widget settings. This function never returns `NULL`; if the icon can’t be rendered (perhaps because an image file fails to load), a default "missing image" icon will be returned instead.

Parameters

icon_set	a GtkIconSet	
context	a GtkStyleContext	
size	icon size (GtkIconSize). A size of `(GtkIconSize)-1` means render at the size of the source and don't scale.	[type int]

Returns

a GdkPixbuf to be displayed.

[transfer full]

Since: 3.0

---

gtk_icon_set_render_icon_surface ()

```
cairo_surface_t *
gtk_icon_set_render_icon_surface (GtkIconSet *icon_set,
                                  GtkStyleContext *context,
                                  GtkIconSize size,
                                  int scale,
                                  GdkWindow *for_window);
```

`gtk_icon_set_render_icon_surface` has been deprecated since version 3.10 and should not be used in newly-written code.

Use GtkIconTheme instead.

Renders an icon using `gtk_render_icon_pixbuf()` and converts it to a cairo surface.

This function never returns `NULL`; if the icon can't be rendered (perhaps because an image file fails to load), a default "missing image" icon will be returned instead.

Parameters

icon_set	a GtkIconSet	
context	a GtkStyleContext	
size	icon size (GtkIconSize). A size of `(GtkIconSize)-1` means render at the size of the source and don't scale.	[type int]
scale	the window scale to render for	
for_window	GdkWindow to optimize drawing for, or `NULL`.	[allow-none]

Returns

a cairo_surface_t to be displayed.

[transfer full]

Since: 3.10

---

gtk_icon_set_unref ()

```
void
gtk_icon_set_unref (GtkIconSet *icon_set);
```

`gtk_icon_set_unref` has been deprecated since version 3.10 and should not be used in newly-written code.

Use GtkIconTheme instead.

Decrements the reference count on `icon_set` , and frees memory if the reference count reaches 0.

Parameters

icon_set	a GtkIconSet

---

gtk_icon_size_lookup ()

```
gboolean
gtk_icon_size_lookup (GtkIconSize size,
                      gint *width,
                      gint *height);
```

Obtains the pixel size of a semantic icon size `size` : GTK_ICON_SIZE_MENU, GTK_ICON_SIZE_BUTTON, etc. This function isn't normally needed, `gtk_icon_theme_load_icon()` is the usual way to get an icon for rendering, then just look at the size of the rendered pixbuf. The rendered pixbuf may not even correspond to the width/height returned by `gtk_icon_size_lookup()`, because themes are free to render the pixbuf however they like, including changing the usual size.

Parameters

size	an icon size (GtkIconSize).	[type int]
width	location to store icon width.	[out][allow-none]
height	location to store icon height.	[out][allow-none]

Returns

`TRUE` if `size` was a valid size

---

gtk_icon_size_lookup_for_settings ()

```
gboolean
gtk_icon_size_lookup_for_settings (GtkSettings *settings,
                                   GtkIconSize size,
                                   gint *width,
                                   gint *height);
```

`gtk_icon_size_lookup_for_settings` has been deprecated since version 3.10 and should not be used in

newly-written code.

Use `gtk_icon_size_lookup()` instead.

Obtains the pixel size of a semantic icon size, possibly modified by user preferences for a particular GtkSettings. Normally `size` would be GTK_ICON_SIZE_MENU, GTK_ICON_SIZE_BUTTON, etc. This function isn't normally needed, `gtk_widget_render_icon_pixbuf()` is the usual way to get an icon for rendering, then just look at the size of the rendered pixbuf. The rendered pixbuf may not even correspond to the width/height returned by `gtk_icon_size_lookup()`, because themes are free to render the pixbuf however they like, including changing the usual size.

Parameters

settings	a GtkSettings object, used to determine which set of user preferences to used.	
size	an icon size (GtkIconSize).	[type int]
width	location to store icon width.	[out][allow-none]
height	location to store icon height.	[out][allow-none]

Returns

`TRUE` if `size` was a valid size

Since: 2.2

---

gtk_icon_size_register ()

```
GtkIconSize
gtk_icon_size_register (const gchar *name,
                        gint width,
                        gint height);
```

`gtk_icon_size_register` has been deprecated since version 3.10 and should not be used in newly-written code.

Use GtkIconTheme instead.

Registers a new icon size, along the same lines as GTK_ICON_SIZE_MENU, etc. Returns the integer value for the size.

Parameters

name	name of the icon size
width	the icon width
height	the icon height

Returns

integer value representing the size (GtkIconSize).

[type int]

---

gtk_icon_size_register_alias ()

```
void
gtk_icon_size_register_alias (const gchar *alias,
                              GtkIconSize target);
```

`gtk_icon_size_register_alias` has been deprecated since version 3.10 and should not be used in newly-written code.

Use GtkIconTheme instead.

Registers `alias` as another name for `target` . So calling `gtk_icon_size_from_name()` with `alias` as argument will return `target` .

Parameters

alias	an alias for `target`
target	an existing icon size (GtkIconSize). [type int]

---

gtk_icon_size_from_name ()

```
GtkIconSize
gtk_icon_size_from_name (const gchar *name);
```

`gtk_icon_size_from_name` has been deprecated since version 3.10 and should not be used in newly-written code.

Use GtkIconTheme instead.

Looks up the icon size associated with `name` .

Parameters

name	the name to look up.

Returns

the icon size (GtkIconSize).

[type int]

---

gtk_icon_size_get_name ()

```
const gchar *
gtk_icon_size_get_name (GtkIconSize size);
```

`gtk_icon_size_get_name` has been deprecated since version 3.10 and should not be used in newly-written

code.

Use GtkIconTheme instead.

Gets the canonical name of the given icon size. The returned string is statically allocated and should not be freed.

Parameters

size	a GtkIconSize.	[type int]

Returns

the name of the given icon size.

---

gtk_icon_set_get_sizes ()

```
void
gtk_icon_set_get_sizes (GtkIconSet *icon_set,
                        GtkIconSize **sizes,
                        gint *n_sizes);
```

`gtk_icon_set_get_sizes` has been deprecated since version 3.10 and should not be used in newly-written code.

Use GtkIconTheme instead.

Obtains a list of icon sizes this icon set can render. The returned array must be freed with `g_free()`.

Parameters

icon_set	a GtkIconSet	
sizes	return location for array of sizes (GtkIconSize).	[array length=n_sizes][out][type int]
n_sizes	location to store number of elements in returned array	

---

gtk_icon_source_get_direction ()

```
GtkTextDirection
gtk_icon_source_get_direction (const GtkIconSource *source);
```

`gtk_icon_source_get_direction` has been deprecated since version 3.10 and should not be used in newly-written code.

Use GtkIconTheme instead.

Obtains the text direction this icon source applies to. The return value is only useful/meaningful if the text direction is not wildcarded.

Parameters

source	a GtkIconSource

Returns

text direction this source matches

---

gtk_icon_source_get_direction_wildcarded ()

```
gboolean
gtk_icon_source_get_direction_wildcarded
                                (const GtkIconSource *source);
```

`gtk_icon_source_get_direction_wildcarded` has been deprecated since version 3.10 and should not be used in newly-written code.

Use GtkIconTheme instead.

Gets the value set by `gtk_icon_source_set_direction_wildcarded()`.

Parameters

source	a GtkIconSource

Returns

`TRUE` if this icon source is a base for any text direction variant

---

gtk_icon_source_get_filename ()

```
const gchar *
gtk_icon_source_get_filename (const GtkIconSource *source);
```

`gtk_icon_source_get_filename` has been deprecated since version 3.10 and should not be used in newly-written code.

Use GtkIconTheme instead.

Retrieves the source filename, or `NULL` if none is set. The filename is not a copy, and should not be modified or expected to persist beyond the lifetime of the icon source.

Parameters

source	a GtkIconSource

Returns

image filename. This string must not be modified or freed.

[type filename]

---

gtk_icon_source_get_pixbuf ()

```
GdkPixbuf *
gtk_icon_source_get_pixbuf (const GtkIconSource *source);
```

`gtk_icon_source_get_pixbuf` has been deprecated since version 3.10 and should not be used in newly-written code.

Use GtkIconTheme instead.

Retrieves the source pixbuf, or `NULL` if none is set. In addition, if a filename source is in use, this function in some cases will return the pixbuf from loaded from the filename. This is, for example, true for the GtkIconSource passed to the GtkStyle `render_icon()` virtual function. The reference count on the pixbuf is not incremented.

Parameters

source	a GtkIconSource

Returns

source pixbuf.

[transfer none]

---

gtk_icon_source_get_icon_name ()

```
const gchar *
gtk_icon_source_get_icon_name (const GtkIconSource *source);
```

`gtk_icon_source_get_icon_name` has been deprecated since version 3.10 and should not be used in newly-written code.

Use GtkIconTheme instead.

Retrieves the source icon name, or `NULL` if none is set. The icon_name is not a copy, and should not be modified or expected to persist beyond the lifetime of the icon source.

Parameters

source	a GtkIconSource

Returns

icon name. This string must not be modified or freed.

---

gtk_icon_source_get_size ()

```
GtkIconSize
gtk_icon_source_get_size (const GtkIconSource *source);
```

`gtk_icon_source_get_size` has been deprecated since version 3.10 and should not be used in newly-written code.

Use GtkIconTheme instead.

Obtains the icon size this source applies to. The return value is only useful/meaningful if the icon size is not wildcarded.

Parameters

source	a GtkIconSource

Returns

icon size (GtkIconSize) this source matches.

[type int]

---

gtk_icon_source_get_size_wildcarded ()

```
gboolean
gtk_icon_source_get_size_wildcarded (const GtkIconSource *source);
```

`gtk_icon_source_get_size_wildcarded` has been deprecated since version 3.10 and should not be used in newly-written code.

Use GtkIconTheme instead.

Gets the value set by `gtk_icon_source_set_size_wildcarded()`.

Parameters

source	a GtkIconSource

Returns

`TRUE` if this icon source is a base for any icon size variant

---

gtk_icon_source_get_state ()

```
GtkStateType
gtk_icon_source_get_state (const GtkIconSource *source);
```

`gtk_icon_source_get_state` has been deprecated since version 3.10 and should not be used in newly-written code.

Use GtkIconTheme instead.

Obtains the widget state this icon source applies to. The return value is only useful/meaningful if the widget state is not wildcarded.

Parameters

source	a GtkIconSource

Returns

widget state this source matches

---

gtk_icon_source_get_state_wildcarded ()

```
gboolean
gtk_icon_source_get_state_wildcarded (const GtkIconSource *source);
```

`gtk_icon_source_get_state_wildcarded` has been deprecated since version 3.10 and should not be used in newly-written code.

Use GtkIconTheme instead.

Gets the value set by `gtk_icon_source_set_state_wildcarded()`.

Parameters

source	a GtkIconSource

Returns

`TRUE` if this icon source is a base for any widget state variant

---

gtk_icon_source_new ()

```
GtkIconSource *
gtk_icon_source_new (void);
```

`gtk_icon_source_new` has been deprecated since version 3.10 and should not be used in newly-written code.

Use GtkIconTheme instead.

Creates a new GtkIconSource. A GtkIconSource contains a GdkPixbuf (or image filename) that serves as the base image for one or more of the icons in a GtkIconSet, along with a specification for which icons in the icon set will be based on that pixbuf or image file. An icon set contains a set of icons that represent “the same” logical concept in different states, different global text directions, and different sizes.

So for example a web browser’s “Back to Previous Page” icon might point in a different direction in Hebrew and in English; it might look different when insensitive; and it might change size depending on toolbar mode (small/large icons). So a single icon set would contain all those variants of the icon. GtkIconSet contains a list of GtkIconSource from which it can derive specific icon variants in the set.

In the simplest case, GtkIconSet contains one source pixbuf from which it derives all variants. The convenience function `gtk_icon_set_new_from_pixbuf()` handles this case; if you only have one source pixbuf, just use that function.

If you want to use a different base pixbuf for different icon variants, you create multiple icon sources, mark which variants they'll be used to create, and add them to the icon set with `gtk_icon_set_add_source()`.

By default, the icon source has all parameters wildcarded. That is, the icon source will be used as the base icon for any desired text direction, widget state, or icon size.

Returns

a new GtkIconSource

---

gtk_icon_source_set_direction ()

```
void
gtk_icon_source_set_direction (GtkIconSource *source,
                               GtkTextDirection direction);
```

`gtk_icon_source_set_direction` has been deprecated since version 3.10 and should not be used in newly-written code.

Use GtkIconTheme instead.

Sets the text direction this icon source is intended to be used with.

Setting the text direction on an icon source makes no difference if the text direction is wildcarded. Therefore, you should usually call `gtk_icon_source_set_direction_wildcarded()` to un-wildcard it in addition to calling this function.

Parameters

source	a GtkIconSource
direction	text direction this source applies to

---

gtk_icon_source_set_direction_wildcarded ()

```
void
gtk_icon_source_set_direction_wildcarded
                               (GtkIconSource *source,
                                gboolean setting);
```

`gtk_icon_source_set_direction_wildcarded` has been deprecated since version 3.10 and should not be used in newly-written code.

Use GtkIconTheme instead.

If the text direction is wildcarded, this source can be used as the base image for an icon in any GtkTextDirection. If the text direction is not wildcarded, then the text direction the icon source applies to should be set with `gtk_icon_source_set_direction()`, and the icon source will only be used with that text

direction.

GtkIconSet prefers non-wildcarded sources (exact matches) over wildcarded sources, and will use an exact match when possible.

Parameters

source	a GtkIconSource
setting	`TRUE` to wildcard the text direction

---

gtk_icon_source_set_filename ()

```
void
gtk_icon_source_set_filename (GtkIconSource *source,
                              const gchar *filename);
```

`gtk_icon_source_set_filename` has been deprecated since version 3.10 and should not be used in newly-written code.

Use GtkIconTheme instead.

Sets the name of an image file to use as a base image when creating icon variants for GtkIconSet. The filename must be absolute.

Parameters

source	a GtkIconSource	
filename	image file to use.	[type filename]

---

gtk_icon_source_set_pixbuf ()

```
void
gtk_icon_source_set_pixbuf (GtkIconSource *source,
                            GdkPixbuf *pixbuf);
```

`gtk_icon_source_set_pixbuf` has been deprecated since version 3.10 and should not be used in newly-written code.

Use GtkIconTheme instead.

Sets a pixbuf to use as a base image when creating icon variants for GtkIconSet.

Parameters

source	a GtkIconSource
pixbuf	pixbuf to use as a source

---

gtk_icon_source_set_icon_name ()

```
void
gtk_icon_source_set_icon_name (GtkIconSource *source,
                               const gchar *icon_name);
```

`gtk_icon_source_set_icon_name` has been deprecated since version 3.10 and should not be used in newly-written code.

Use GtkIconTheme instead.

Sets the name of an icon to look up in the current icon theme to use as a base image when creating icon variants for GtkIconSet.

Parameters

source	a GtkIconSource	
icon_name	name of icon to use.	[allow-none]

---

gtk_icon_source_set_size ()

```
void
gtk_icon_source_set_size (GtkIconSource *source,
                          GtkIconSize size);
```

`gtk_icon_source_set_size` has been deprecated since version 3.10 and should not be used in newly-written code.

Use GtkIconTheme instead.

Sets the icon size this icon source is intended to be used with.

Setting the icon size on an icon source makes no difference if the size is wildcarded. Therefore, you should usually call `gtk_icon_source_set_size_wildcarded()` to un-wildcard it in addition to calling this function.

Parameters

source	a GtkIconSource	
size	icon size (GtkIconSize) this source applies to.	[type int]

---

gtk_icon_source_set_size_wildcarded ()

```
void
gtk_icon_source_set_size_wildcarded (GtkIconSource *source,
                                     gboolean setting);
```

`gtk_icon_source_set_size_wildcarded` has been deprecated since version 3.10 and should not be used in newly-written code.

Use GtkIconTheme instead.

If the icon size is wildcarded, this source can be used as the base image for an icon of any size. If the size is not

wildcarded, then the size the source applies to should be set with `gtk_icon_source_set_size()` and the icon source will only be used with that specific size.

GtkIconSet prefers non-wildcarded sources (exact matches) over wildcarded sources, and will use an exact match when possible.

GtkIconSet will normally scale wildcarded source images to produce an appropriate icon at a given size, but will not change the size of source images that match exactly.

Parameters

source	a GtkIconSource
setting	`TRUE` to wildcard the widget state

---

gtk_icon_source_set_state ()

```
void
gtk_icon_source_set_state (GtkIconSource *source,
                           GtkStateType state);
```

`gtk_icon_source_set_state` has been deprecated since version 3.10 and should not be used in newly-written code.

Use GtkIconTheme instead.

Sets the widget state this icon source is intended to be used with.

Setting the widget state on an icon source makes no difference if the state is wildcarded. Therefore, you should usually call `gtk_icon_source_set_state_wildcarded()` to un-wildcard it in addition to calling this function.

Parameters

source	a GtkIconSource
state	widget state this source applies to

---

gtk_icon_source_set_state_wildcarded ()

```
void
gtk_icon_source_set_state_wildcarded (GtkIconSource *source,
                                      gboolean setting);
```

`gtk_icon_source_set_state_wildcarded` has been deprecated since version 3.10 and should not be used in newly-written code.

Use GtkIconTheme instead.

If the widget state is wildcarded, this source can be used as the base image for an icon in any GtkStateType. If the widget state is not wildcarded, then the state the source applies to should be set with `gtk_icon_source_set_state()` and the icon source will only be used with that specific state.

GtkIconSet prefers non-wildcarded sources (exact matches) over wildcarded sources, and will use an exact match when possible.

GtkIconSet will normally transform wildcarded source images to produce an appropriate icon for a given state, for example lightening an image on prelight, but will not modify source images that match exactly.

Parameters

source	a GtkIconSource
setting	TRUE to wildcard the widget state

Types and Values

GtkIconSource

```
typedef struct _GtkIconSource GtkIconSource;
```

---

struct GtkIconFactory

```
struct GtkIconFactory;
```

---

struct GtkIconFactoryClass

```
struct GtkIconFactoryClass {
  GObjectClass parent_class;
};
```

Members

GtkIconSet

```
typedef struct _GtkIconSet GtkIconSet;
```

---

enum GtkIconSize

Built-in stock icon sizes.

Members

GTK_ICON_SIZE_INVALID	Invalid size.
GTK_ICON_SIZE_MENU	Size appropriate for menus (16px).
GTK_ICON_SIZE_SMALL_TOO	Size appropriate for small toolbars

LBAR	(16px).
GTK_ICON_SIZE_LARGE_TOOL BAR	Size appropriate for large toolbars (24px)
GTK_ICON_SIZE_BUTTON	Size appropriate for buttons (16px)
GTK_ICON_SIZE_DND	Size appropriate for drag and drop (32px)
GTK_ICON_SIZE_DIALOG	Size appropriate for dialogs (48px)

GtkNumerableIcon

GtkNumerableIcon — A GIcon that allows numbered emblems

Functions

GIcon *	gtk_numerable_icon_new ()
GIcon *	gtk_numerable_icon_new_with_style_context ()
GIcon *	gtk_numerable_icon_get_background_gicon ()
void	gtk_numerable_icon_set_background_gicon ()
const gchar *	gtk_numerable_icon_get_background_icon_name ()
void	gtk_numerable_icon_set_background_icon_name ()
gint	gtk_numerable_icon_get_count ()
void	gtk_numerable_icon_set_count ()
const gchar *	gtk_numerable_icon_get_label ()
void	gtk_numerable_icon_set_label ()
GtkStyleContext *	gtk_numerable_icon_get_style_context ()
void	gtk_numerable_icon_set_style_context ()

Properties

GIcon *	background-icon	Read / Write
gchar *	background-icon-name	Read / Write
gint	count	Read / Write
gchar *	label	Read / Write
GtkStyleContext *	style-context	Read / Write

Types and Values

struct	GtkNumerableIcon

Object Hierarchy

```
GObject
└── GEmblemedIcon
    └── GtkNumerableIcon
```

Implemented Interfaces

GtkNumerableIcon implements GIcon.

Includes

```
#include <gtk/gtk.h>
```

Description

GtkNumerableIcon is a subclass of GEmblemedIcon that can show a number or short string as an emblem. The number can be overlayed on top of another emblem, if desired.

It supports theming by taking font and color information from a provided GtkStyleContext; see `gtk_numerable_icon_set_style_context()`.

Typical numerable icons: 

Functions

gtk_numerable_icon_new ()

```
GIcon *
gtk_numerable_icon_new (GIcon *base_icon);
```

`gtk_numerable_icon_new` has been deprecated since version 3.14 and should not be used in newly-written code.

Creates a new unthemed GtkNumerableIcon.

Parameters

base_icon	a GIcon to overlay on

Returns

a new GIcon.

[transfer full]

Since: 3.0

---

gtk_numerable_icon_new_with_style_context ()

```
GIcon *
gtk_numerable_icon_new_with_style_context
```

```
                                (GIcon *base_icon,
                                 GtkStyleContext *context);
```

`gtk_numerable_icon_new_with_style_context` has been deprecated since version 3.14 and should not be used in newly-written code.

Creates a new GtkNumerableIcon which will themed according to the passed GtkStyleContext. This is a convenience constructor that calls `gtk_numerable_icon_set_style_context()` internally.

Parameters

base_icon	a GIcon to overlay on
context	a GtkStyleContext

Returns

a new GIcon.

[transfer full]

Since: 3.0

---

gtk_numerable_icon_get_background_gicon ()

```
GIcon *
gtk_numerable_icon_get_background_gicon
                                (GtkNumerableIcon *self);
```

`gtk_numerable_icon_get_background_gicon` has been deprecated since version 3.14 and should not be used in newly-written code.

Returns the GIcon that was set as the base background image, or `NULL` if there's none. The caller of this function does not own a reference to the returned GIcon.

Parameters

self	a GtkNumerableIcon

Returns

a GIcon, or `NULL`.

[nullable][transfer none]

Since: 3.0

---

gtk_numerable_icon_set_background_gicon ()

```
void
gtk_numerable_icon_set_background_gicon
                                (GtkNumerableIcon *self,
```

```
                                GIcon *icon);
```

`gtk_numerable_icon_set_background_gicon` has been deprecated since version 3.14 and should not be used in newly-written code.

Updates the icon to use `icon` as the base background image. If `icon` is `NULL`, `self` will go back using style information or default theming for its background image.

If this method is called and an icon name was already set as background for the icon, `icon` will be used, i.e. the last method called between `gtk_numerable_icon_set_background_gicon()` and `gtk_numerable_icon_set_background_icon_name()` has always priority.

Parameters

self	a GtkNumerableIcon	
icon	a GIcon, or `NULL`.	[allow-none]

Since: 3.0

---

gtk_numerable_icon_get_background_icon_name ()

```
const gchar *
gtk_numerable_icon_get_background_icon_name
                                (GtkNumerableIcon *self);
```

`gtk_numerable_icon_get_background_icon_name` has been deprecated since version 3.14 and should not be used in newly-written code.

Returns the icon name used as the base background image, or `NULL` if there's none.

Parameters

self	a GtkNumerableIcon

Returns

an icon name, or `NULL`.

[nullable]

Since: 3.0

---

gtk_numerable_icon_set_background_icon_name ()

```
void
gtk_numerable_icon_set_background_icon_name
                                (GtkNumerableIcon *self,
                                 const gchar *icon_name);
```

`gtk_numerable_icon_set_background_icon_name` has been deprecated since version 3.14 and should not be used in newly-written code.

Updates the icon to use the icon named `icon_name` from the current icon theme as the base background image.

If `icon_name` is `NULL`, `self` will go back using style information or default theming for its background image.

If this method is called and a GIcon was already set as background for the icon, `icon_name` will be used, i.e. the last method called between `gtk_numerable_icon_set_background_icon_name()` and `gtk_numerable_icon_set_background_gicon()` has always priority.

Parameters

self	a GtkNumerableIcon	
icon_name	an icon name, or `NULL`.	[allow-none]

Since: 3.0

---

gtk_numerable_icon_get_count ()

```
gint
gtk_numerable_icon_get_count (GtkNumerableIcon *self);
```

`gtk_numerable_icon_get_count` has been deprecated since version 3.14 and should not be used in newly-written code.

Returns the value currently displayed by `self` .

Parameters

self	a GtkNumerableIcon

Returns

the currently displayed value

Since: 3.0

---

gtk_numerable_icon_set_count ()

```
void
gtk_numerable_icon_set_count (GtkNumerableIcon *self,
                              gint count);
```

`gtk_numerable_icon_set_count` has been deprecated since version 3.14 and should not be used in newly-written code.

Sets the currently displayed value of `self` to `count` .

The numeric value is always clamped to make it two digits, i.e. between -99 and 99. Setting a count of zero removes the emblem. If this method is called, and a label was already set on the icon, it will automatically be reset to `NULL` before rendering the number, i.e. the last method called between `gtk_numerable_icon_set_count()` and `gtk_numerable_icon_set_label()` has always priority.

Parameters

self	a GtkNumerableIcon	
count	a number between -99 and 99	

Since: 3.0

---

gtk_numerable_icon_get_label ()

```
const gchar *
gtk_numerable_icon_get_label (GtkNumerableIcon *self);
```

`gtk_numerable_icon_get_label` has been deprecated since version 3.14 and should not be used in newly-written code.

Returns the currently displayed label of the icon, or `NULL`.

Parameters

self	a GtkNumerableIcon	

Returns

the currently displayed label.

[nullable]

Since: 3.0

---

gtk_numerable_icon_set_label ()

```
void
gtk_numerable_icon_set_label (GtkNumerableIcon *self,
                              const gchar *label);
```

`gtk_numerable_icon_set_label` has been deprecated since version 3.14 and should not be used in newly-written code.

Sets the currently displayed value of `self` to the string in `label` . Setting an empty label removes the emblem.

Note that this is meant for displaying short labels, such as roman numbers, or single letters. For roman numbers, consider using the Unicode characters U+2160 - U+217F. Strings longer than two characters will likely not be rendered very well.

If this method is called, and a number was already set on the icon, it will automatically be reset to zero before rendering the label, i.e. the last method called between `gtk_numerable_icon_set_label()` and `gtk_numerable_icon_set_count()` has always priority.

Parameters

self	a GtkNumerableIcon	
label	a short label, or `NULL`.	[allow-none]

Since: 3.0

---

gtk_numerable_icon_get_style_context ()

```
GtkStyleContext *
gtk_numerable_icon_get_style_context (GtkNumerableIcon *self);
```

`gtk_numerable_icon_get_style_context` has been deprecated since version 3.14 and should not be used in newly-written code.

Returns the GtkStyleContext used by the icon for theming, or `NULL` if there's none.

Parameters

self	a GtkNumerableIcon

Returns

a GtkStyleContext, or `NULL`. This object is internal to GTK+ and should not be unreffed. Use `g_object_ref()` if you want to keep it around.

[nullable][transfer none]

Since: 3.0

---

gtk_numerable_icon_set_style_context ()

```
void
gtk_numerable_icon_set_style_context (GtkNumerableIcon *self,
                                      GtkStyleContext *style);
```

`gtk_numerable_icon_set_style_context` has been deprecated since version 3.14 and should not be used in newly-written code.

Updates the icon to fetch theme information from the given GtkStyleContext.

Parameters

self	a GtkNumerableIcon
style	a GtkStyleContext

Since: 3.0

Types and Values

struct GtkNumerableIcon

```
struct GtkNumerableIcon;
```

Property Details

The `"background-icon"` property

```
  "background-icon"          GIcon *
```

The icon for the number emblem background.

Flags: Read / Write

The `"background-icon-name"` property

```
  "background-icon-name"     gchar *
```

The icon name for the number emblem background.

Flags: Read / Write

Default value: NULL

The `"count"` property

```
  "count"                    gint
```

The count of the emblem currently displayed.

Flags: Read / Write

Allowed values: [-99,99]

Default value: 0

The `"label"` property

```
  "label"                    gchar *
```

The label to be displayed over the icon.

Flags: Read / Write

Default value: NULL

The `"style-context"` property

```
  "style-context"            GtkStyleContext *
```

The style context to theme the icon appearance.

Flags: Read / Write

GtkArrow

GtkArrow — Displays an arrow

Functions

GtkWidget *	gtk_arrow_new ()
void	gtk_arrow_set ()

Properties

GtkArrowType	arrow-type	Read / Write
GtkShadowType	shadow-type	Read / Write

Style Properties

gfloat	arrow-scaling	Read

Types and Values

struct	GtkArrow

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkMisc
            └── GtkArrow
```

Implemented Interfaces

GtkArrow implements AtkImplementorIface and GtkBuildable.

Includes

```
#include <gtk/gtk.h>
```

Description

GtkArrow should be used to draw simple arrows that need to point in one of the four cardinal directions (up, down, left, or right). The style of the arrow can be one of shadow in, shadow out, etched in, or etched out. Note that these directions and style types may be amended in versions of GTK+ to come.

GtkArrow will fill any space alloted to it, but since it is inherited from GtkMisc, it can be padded and/or aligned, to fill exactly the space the programmer desires.

Arrows are created with a call to `gtk_arrow_new()`. The direction or style of an arrow can be changed after

creation by using `gtk_arrow_set()`.

GtkArrow has been deprecated; you can simply use a GtkImage with a suitable icon name, such as “pan-down-symbolic“. When replacing GtkArrow by an image, pay attention to the fact that GtkArrow is doing automatic flipping between GTK_ARROW_LEFT and GTK_ARROW_RIGHT, depending on the text direction. To get the same effect with an image, use the icon names “pan-start-symbolic“ and “pan-end-symbolic“, which react to the text direction.

Functions

gtk_arrow_new ()

```
GtkWidget *
gtk_arrow_new (GtkArrowType arrow_type,
               GtkShadowType shadow_type);
```

`gtk_arrow_new` has been deprecated since version 3.14 and should not be used in newly-written code.

Use a GtkImage with a suitable icon.

Creates a new GtkArrow widget.

Parameters

arrow_type	a valid GtkArrowType.
shadow_type	a valid GtkShadowType.

Returns

the new GtkArrow widget.

---

gtk_arrow_set ()

```
void
gtk_arrow_set (GtkArrow *arrow,
               GtkArrowType arrow_type,
               GtkShadowType shadow_type);
```

`gtk_arrow_set` has been deprecated since version 3.14 and should not be used in newly-written code.

Use a GtkImage with a suitable icon.

Sets the direction and style of the GtkArrow, `arrow` .

Parameters

arrow	a widget of type GtkArrow.
arrow_type	a valid GtkArrowType.
shadow_type	a valid GtkShadowType.

Types and Values

struct GtkArrow

```
struct GtkArrow;
```

Property Details

The “arrow-type” property

```
  “arrow-type”               GtkArrowType
```

The direction the arrow should point.

Flags: Read / Write

Default value: GTK_ARROW_RIGHT

The “shadow-type” property

```
  “shadow-type”              GtkShadowType
```

Appearance of the shadow surrounding the arrow.

Flags: Read / Write

Default value: GTK_SHADOW_OUT

Style Property Details

The “arrow-scaling” style property

```
  “arrow-scaling”            gfloat
```

Amount of space used up by arrow.

Flags: Read

Allowed values: [0,1]

Default value: 0.7

See Also

gtk_render_arrow()

GtkStatusIcon

GtkStatusIcon — Display an icon in the system tray

Functions

GtkStatusIcon *	gtk_status_icon_new ()
GtkStatusIcon *	gtk_status_icon_new_from_pixbuf ()
GtkStatusIcon *	gtk_status_icon_new_from_file ()
GtkStatusIcon *	gtk_status_icon_new_from_stock ()
GtkStatusIcon *	gtk_status_icon_new_from_icon_name ()
GtkStatusIcon *	gtk_status_icon_new_from_gicon ()
void	gtk_status_icon_set_from_pixbuf ()
void	gtk_status_icon_set_from_file ()
void	gtk_status_icon_set_from_stock ()
void	gtk_status_icon_set_from_icon_name ()
void	gtk_status_icon_set_from_gicon ()
GtkImageType	gtk_status_icon_get_storage_type ()
GdkPixbuf *	gtk_status_icon_get_pixbuf ()
const gchar *	gtk_status_icon_get_stock ()
const gchar *	gtk_status_icon_get_icon_name ()
GIcon *	gtk_status_icon_get_gicon ()
gint	gtk_status_icon_get_size ()
void	gtk_status_icon_set_screen ()
GdkScreen *	gtk_status_icon_get_screen ()
void	gtk_status_icon_set_tooltip_text ()
gchar *	gtk_status_icon_get_tooltip_text ()
void	gtk_status_icon_set_tooltip_markup ()
gchar *	gtk_status_icon_get_tooltip_markup ()
void	gtk_status_icon_set_has_tooltip ()
gboolean	gtk_status_icon_get_has_tooltip ()
void	gtk_status_icon_set_title ()
const gchar *	gtk_status_icon_get_title ()
void	gtk_status_icon_set_name ()
void	gtk_status_icon_set_visible ()
gboolean	gtk_status_icon_get_visible ()
gboolean	gtk_status_icon_is_embedded ()
void	gtk_status_icon_position_menu ()
gboolean	gtk_status_icon_get_geometry ()
guint32	gtk_status_icon_get_x11_window_id ()

Properties

gboolean	embedded	Read
gchar *	file	Write
GIcon *	gicon	Read / Write
gboolean	has-tooltip	Read / Write
gchar *	icon-name	Read / Write
GtkOrientation	orientation	Read
GdkPixbuf *	pixbuf	Read / Write

GdkScreen *	screen	Read / Write
gint	size	Read
gchar *	stock	Read / Write
GtkImageType	storage-type	Read
gchar *	title	Read / Write
gchar *	tooltip-markup	Read / Write
gchar *	tooltip-text	Read / Write
gboolean	visible	Read / Write

Signals

void	activate	Action
gboolean	button-press-event	Run Last
gboolean	button-release-event	Run Last
void	popup-menu	Action
gboolean	query-tooltip	Run Last
gboolean	scroll-event	Run Last
gboolean	size-changed	Run Last

Types and Values

struct	GtkStatusIcon

Object Hierarchy

```
GObject
└── GtkStatusIcon
```

Includes

```
#include <gtk/gtk.h>
```

Description

The “system tray” or notification area is normally used for transient icons that indicate some special state. For example, a system tray icon might appear to tell the user that they have new mail, or have an incoming instant message, or something along those lines. The basic idea is that creating an icon in the notification area is less annoying than popping up a dialog.

A GtkStatusIcon object can be used to display an icon in a “system tray”. The icon can have a tooltip, and the user can interact with it by activating it or popping up a context menu.

It is very important to notice that status icons depend on the existence of a notification area being available to the user; you should not use status icons as the only way to convey critical information regarding your application, as the notification area may not exist on the user's environment, or may have been removed. You should always check that a status icon has been embedded into a notification area by using `gtk_status_icon_is_embedded()`, and gracefully recover if the function returns `FALSE`.

On X11, the implementation follows the FreeDesktop System Tray Specification. Implementations of the “tray” side of this specification can be found e.g. in the GNOME 2 and KDE panel applications.

Note that a GtkStatusIcon is not a widget, but just a GObject. Making it a widget would be impractical, since

the system tray on Windows doesn’t allow to embed arbitrary widgets.

GtkStatusIcon has been deprecated in 3.14. You should consider using notifications or more modern platform-specific APIs instead. GLib provides the GNotification API which works well with GtkApplication on multiple platforms and environments, and should be the preferred mechanism to notify the users of transient status updates. See this HowDoI for code examples.

Functions

gtk_status_icon_new ()

```
GtkStatusIcon *
gtk_status_icon_new (void);
```

`gtk_status_icon_new` has been deprecated since version 3.14 and should not be used in newly-written code.

Use GNotification and GtkApplication to provide status notifications

Creates an empty status icon object.

Returns

a new GtkStatusIcon

Since: 2.10

---

gtk_status_icon_new_from_pixbuf ()

```
GtkStatusIcon *
gtk_status_icon_new_from_pixbuf (GdkPixbuf *pixbuf);
```

`gtk_status_icon_new_from_pixbuf` has been deprecated since version 3.14 and should not be used in newly-written code.

Use GNotification and GtkApplication to provide status notifications

Creates a status icon displaying `pixbuf` .

The image will be scaled down to fit in the available space in the notification area, if necessary.

Parameters

pixbuf	a GdkPixbuf

Returns

a new GtkStatusIcon

Since: 2.10

---

gtk_status_icon_new_from_file ()

```
GtkStatusIcon *
gtk_status_icon_new_from_file (const gchar *filename);
```

`gtk_status_icon_new_from_file` has been deprecated since version 3.14 and should not be used in newly-written code.

Use GNotification and GtkApplication to provide status notifications

Creates a status icon displaying the file `filename` .

The image will be scaled down to fit in the available space in the notification area, if necessary.

Parameters

filename	a filename.	[type filename]

Returns

a new GtkStatusIcon

Since: 2.10

---

gtk_status_icon_new_from_stock ()

```
GtkStatusIcon *
gtk_status_icon_new_from_stock (const gchar *stock_id);
```

`gtk_status_icon_new_from_stock` has been deprecated since version 3.14 and should not be used in newly-written code.

Use GNotification and GtkApplication to provide status notifications

Creates a status icon displaying a stock icon. Sample stock icon names are GTK_STOCK_OPEN, GTK_STOCK_QUIT. You can register your own stock icon names, see `gtk_icon_factory_add_default()` and `gtk_icon_factory_add()`.

Parameters

stock_id	a stock icon id

Returns

a new GtkStatusIcon

Since: 2.10

---

gtk_status_icon_new_from_icon_name ()

```
GtkStatusIcon *
gtk_status_icon_new_from_icon_name (const gchar *icon_name);
```

`gtk_status_icon_new_from_icon_name` has been deprecated since version 3.14 and should not be used in newly-written code.

Use GNotification and GtkApplication to provide status notifications

Creates a status icon displaying an icon from the current icon theme. If the current icon theme is changed, the icon will be updated appropriately.

Parameters

icon_name	an icon name

Returns

a new GtkStatusIcon

Since: 2.10

---

gtk_status_icon_new_from_gicon ()

```
GtkStatusIcon *
gtk_status_icon_new_from_gicon (GIcon *icon);
```

`gtk_status_icon_new_from_gicon` has been deprecated since version 3.14 and should not be used in newly-written code.

Use GNotification and GtkApplication to provide status notifications

Creates a status icon displaying a GIcon. If the icon is a themed icon, it will be updated when the theme changes.

Parameters

icon	a GIcon

Returns

a new GtkStatusIcon

Since: 2.14

---

gtk_status_icon_set_from_pixbuf ()

```
void
gtk_status_icon_set_from_pixbuf (GtkStatusIcon *status_icon,
                                 GdkPixbuf *pixbuf);
```

`gtk_status_icon_set_from_pixbuf` has been deprecated since version 3.14 and should not be used in newly-written code.

Use GNotification and GtkApplication to provide status notifications; you can use `g_notification_set_icon()` to associate a GIcon with a notification

Makes `status_icon` display `pixbuf` . See `gtk_status_icon_new_from_pixbuf()` for details.

Parameters

status_icon	a GtkStatusIcon	
pixbuf	a GdkPixbuf or `NULL`.	[allow-none]

Since: 2.10

---

gtk_status_icon_set_from_file ()

```
void
gtk_status_icon_set_from_file (GtkStatusIcon *status_icon,
                               const gchar *filename);
```

`gtk_status_icon_set_from_file` has been deprecated since version 3.14 and should not be used in newly-written code.

Use GNotification and GtkApplication to provide status notifications; you can use `g_notification_set_icon()` to associate a GIcon with a notification

Makes `status_icon` display the file `filename` . See `gtk_status_icon_new_from_file()` for details.

Parameters

status_icon	a GtkStatusIcon	
filename	a filename.	[type filename]

Since: 2.10

---

gtk_status_icon_set_from_stock ()

```
void
gtk_status_icon_set_from_stock (GtkStatusIcon *status_icon,
                                const gchar *stock_id);
```

`gtk_status_icon_set_from_stock` has been deprecated since version 3.10 and should not be used in newly-written code.

Use `gtk_status_icon_set_from_icon_name()` instead.

Makes `status_icon` display the stock icon with the id `stock_id` . See `gtk_status_icon_new_from_stock()` for details.

Parameters

status_icon	a GtkStatusIcon
stock_id	a stock icon id

Since: 2.10

---

gtk_status_icon_set_from_icon_name ()

```
void
gtk_status_icon_set_from_icon_name (GtkStatusIcon *status_icon,
                                    const gchar *icon_name);
```

`gtk_status_icon_set_from_icon_name` has been deprecated since version 3.14 and should not be used in newly-written code.

Use GNotification and GtkApplication to provide status notifications; you can use `g_notification_set_icon()` to associate a GIcon with a notification

Makes `status_icon` display the icon named `icon_name` from the current icon theme. See `gtk_status_icon_new_from_icon_name()` for details.

Parameters

status_icon	a GtkStatusIcon
icon_name	an icon name

Since: 2.10

---

gtk_status_icon_set_from_gicon ()

```
void
gtk_status_icon_set_from_gicon (GtkStatusIcon *status_icon,
                                GIcon *icon);
```

`gtk_status_icon_set_from_gicon` has been deprecated since version 3.14 and should not be used in newly-written code.

Use GNotification and GtkApplication to provide status notifications; you can use `g_notification_set_icon()` to associate a GIcon with a notification

Makes `status_icon` display the GIcon. See `gtk_status_icon_new_from_gicon()` for details.

Parameters

status_icon	a GtkStatusIcon
icon	a GIcon

Since: 2.14

---

gtk_status_icon_get_storage_type ()

```
GtkImageType
gtk_status_icon_get_storage_type (GtkStatusIcon *status_icon);
```

`gtk_status_icon_get_storage_type` has been deprecated since version 3.14 and should not be used in newly-written code.

Use GNotification and GtkApplication to provide status notifications; there is no direct replacement for this function, and GNotification only supports GIcon instances

Gets the type of representation being used by the GtkStatusIcon to store image data. If the GtkStatusIcon has no image data, the return value will be `GTK_IMAGE_EMPTY`.

Parameters

status_icon	a GtkStatusIcon

Returns

the image representation being used

Since: 2.10

---

gtk_status_icon_get_pixbuf ()

```
GdkPixbuf *
gtk_status_icon_get_pixbuf (GtkStatusIcon *status_icon);
```

`gtk_status_icon_get_pixbuf` has been deprecated since version 3.14 and should not be used in newly-written code.

Use GNotification and GtkApplication to provide status notifications; there is no direct replacement for this function

Gets the GdkPixbuf being displayed by the GtkStatusIcon. The storage type of the status icon must be `GTK_IMAGE_EMPTY` or `GTK_IMAGE_PIXBUF` (see `gtk_status_icon_get_storage_type()`). The caller of this function does not own a reference to the returned pixbuf.

Parameters

status_icon	a GtkStatusIcon

Returns

the displayed pixbuf, or `NULL` if the image is empty.

[nullable][transfer none]

Since: 2.10

---

gtk_status_icon_get_stock ()

```
const gchar *
gtk_status_icon_get_stock (GtkStatusIcon *status_icon);
```

`gtk_status_icon_get_stock` has been deprecated since version 3.10 and should not be used in newly-written code.

Use `gtk_status_icon_get_icon_name()` instead.

Gets the id of the stock icon being displayed by the GtkStatusIcon. The storage type of the status icon must be `GTK_IMAGE_EMPTY` or `GTK_IMAGE_STOCK` (see `gtk_status_icon_get_storage_type()`). The returned string is owned by the GtkStatusIcon and should not be freed or modified.

Parameters

status_icon	a GtkStatusIcon

Returns

stock id of the displayed stock icon, or `NULL` if the image is empty.

[nullable]

Since: 2.10

---

gtk_status_icon_get_icon_name ()

```
const gchar *
gtk_status_icon_get_icon_name (GtkStatusIcon *status_icon);
```

`gtk_status_icon_get_icon_name` has been deprecated since version 3.14 and should not be used in newly-written code.

Use GNotification and GtkApplication to provide status notifications; there is no direct replacement for this function

Gets the name of the icon being displayed by the GtkStatusIcon. The storage type of the status icon must be `GTK_IMAGE_EMPTY` or `GTK_IMAGE_ICON_NAME` (see `gtk_status_icon_get_storage_type()`). The returned string is owned by the GtkStatusIcon and should not be freed or modified.

Parameters

status_icon	a GtkStatusIcon

Returns

name of the displayed icon, or `NULL` if the image is empty.

[nullable]

Since: 2.10

---

gtk_status_icon_get_gicon ()

```
GIcon *
gtk_status_icon_get_gicon (GtkStatusIcon *status_icon);
```

`gtk_status_icon_get_gicon` has been deprecated since version 3.14 and should not be used in newly-written code.

Use GNotification and GtkApplication to provide status notifications; there is no direct replacement for this function

Retrieves the GIcon being displayed by the GtkStatusIcon. The storage type of the status icon must be `GTK_IMAGE_EMPTY` or `GTK_IMAGE_GICON` (see `gtk_status_icon_get_storage_type()`). The caller of this function does not own a reference to the returned GIcon.

If this function fails, `icon` is left unchanged;

Parameters

status_icon	a GtkStatusIcon

Returns

the displayed icon, or `NULL` if the image is empty.

[nullable][transfer none]

Since: 2.14

---

gtk_status_icon_get_size ()

```
gint
gtk_status_icon_get_size (GtkStatusIcon *status_icon);
```

`gtk_status_icon_get_size` has been deprecated since version 3.14 and should not be used in newly-written code.

Use GNotification and GtkApplication to provide status notifications; there is no direct replacement for this function, as the representation of a notification is left to the platform

Gets the size in pixels that is available for the image. Stock icons and named icons adapt their size automatically if the size of the notification area changes. For other storage types, the size-changed signal can be used to react to size changes.

Note that the returned size is only meaningful while the status icon is embedded (see `gtk_status_icon_is_embedded()`).

Parameters

status_icon	a GtkStatusIcon

Returns

the size that is available for the image

Since: 2.10

---

gtk_status_icon_set_screen ()

```
void
gtk_status_icon_set_screen (GtkStatusIcon *status_icon,
                            GdkScreen *screen);
```

`gtk_status_icon_set_screen` has been deprecated since version 3.14 and should not be used in newly-written code.

Use GNotification and GtkApplication to provide status notifications; there is no direct replacement for this function, as GTK typically only has one GdkScreen and notifications are managed by the platform

Sets the GdkScreen where `status_icon` is displayed; if the icon is already mapped, it will be unmapped, and then remapped on the new screen.

Parameters

status_icon	a GtkStatusIcon
screen	a GdkScreen

Since: 2.12

---

gtk_status_icon_get_screen ()

```
GdkScreen *
gtk_status_icon_get_screen (GtkStatusIcon *status_icon);
```

`gtk_status_icon_get_screen` has been deprecated since version 3.14 and should not be used in newly-written code.

Use GNotification and GtkApplication to provide status notifications; there is no direct replacement for this function, as notifications are managed by the platform

Returns the GdkScreen associated with `status_icon` .

Parameters

status_icon	a GtkStatusIcon

Returns

a GdkScreen.

[transfer none]

Since: 2.12

---

gtk_status_icon_set_tooltip_text ()

```
void
gtk_status_icon_set_tooltip_text (GtkStatusIcon *status_icon,
                                  const gchar *text);
```

`gtk_status_icon_set_tooltip_text` has been deprecated since version 3.14 and should not be used in newly-written code.

Use GNotification and GtkApplication to provide status notifications; there is no direct replacement for this function

Sets `text` as the contents of the tooltip.

This function will take care of setting "has-tooltip" to `TRUE` and of the default handler for the "query-tooltip" signal.

See also the "tooltip-text" property and `gtk_tooltip_set_text()`.

Parameters

status_icon	a GtkStatusIcon
text	the contents of the tooltip for `status_icon`

Since: 2.16

---

gtk_status_icon_get_tooltip_text ()

```
gchar *
gtk_status_icon_get_tooltip_text (GtkStatusIcon *status_icon);
```

`gtk_status_icon_get_tooltip_text` has been deprecated since version 3.14 and should not be used in newly-written code.

Use GNotification and GtkApplication to provide status notifications; there is no direct replacement for this function

Gets the contents of the tooltip for `status_icon` .

Parameters

status_icon	a GtkStatusIcon

Returns

the tooltip text, or `NULL`. You should free the returned string with `g_free()` when done.

[nullable]

Since: 2.16

---

gtk_status_icon_set_tooltip_markup ()

```
void
gtk_status_icon_set_tooltip_markup (GtkStatusIcon *status_icon,
                                    const gchar *markup);
```

`gtk_status_icon_set_tooltip_markup` has been deprecated since version 3.14 and should not be used in newly-written code.

Use GNotification and GtkApplication to provide status notifications; there is no direct replacement for this function

Sets `markup` as the contents of the tooltip, which is marked up with the Pango text markup language.

This function will take care of setting “has-tooltip” to `TRUE` and of the default handler for the “query-tooltip” signal.

See also the “tooltip-markup” property and `gtk_tooltip_set_markup()`.

Parameters

status_icon	a GtkStatusIcon	
markup	the contents of the tooltip for `status_icon` , or `NULL`.	[allow-none]

Since: 2.16

---

gtk_status_icon_get_tooltip_markup ()

```
gchar *
gtk_status_icon_get_tooltip_markup (GtkStatusIcon *status_icon);
```

`gtk_status_icon_get_tooltip_markup` has been deprecated since version 3.14 and should not be used in newly-written code.

Use GNotification and GtkApplication to provide status notifications; there is no direct replacement for this function

Gets the contents of the tooltip for `status_icon` .

Parameters

status_icon	a GtkStatusIcon

Returns

the tooltip text, or `NULL`. You should free the returned string with `g_free()` when done.

[nullable]

Since: 2.16

---

gtk_status_icon_set_has_tooltip ()

```
void
gtk_status_icon_set_has_tooltip (GtkStatusIcon *status_icon,
                                 gboolean has_tooltip);
```

`gtk_status_icon_set_has_tooltip` has been deprecated since version 3.14 and should not be used in newly-written code.

Use GNotification and GtkApplication to provide status notifications; there is no direct replacement for this function, but notifications can display an arbitrary amount of text using `g_notification_set_body()`

Sets the has-tooltip property on `status_icon` to `has_tooltip` . See "has-tooltip" for more information.

Parameters

status_icon	a GtkStatusIcon
has_tooltip	whether or not `status_icon` has a tooltip

Since: 2.16

---

gtk_status_icon_get_has_tooltip ()

```
gboolean
gtk_status_icon_get_has_tooltip (GtkStatusIcon *status_icon);
```

`gtk_status_icon_get_has_tooltip` has been deprecated since version 3.14 and should not be used in newly-written code.

Use GNotification and GtkApplication to provide status notifications; there is no direct replacement for this function

Returns the current value of the has-tooltip property. See "has-tooltip" for more information.

Parameters

status_icon	a GtkStatusIcon

Returns

current value of has-tooltip on `status_icon` .

Since: 2.16

---

gtk_status_icon_set_title ()

```
void
gtk_status_icon_set_title (GtkStatusIcon *status_icon,
                           const gchar *title);
```

`gtk_status_icon_set_title` has been deprecated since version 3.14 and should not be used in newly-written code.

Use GNotification and GtkApplication to provide status notifications; you should use `g_notification_set_title()` and `g_notification_set_body()` to present text inside your notification

Sets the title of this tray icon. This should be a short, human-readable, localized string describing the tray icon. It may be used by tools like screen readers to render the tray icon.

Parameters

status_icon	a GtkStatusIcon
title	the title

Since: 2.18

---

gtk_status_icon_get_title ()

```
const gchar *
gtk_status_icon_get_title (GtkStatusIcon *status_icon);
```

`gtk_status_icon_get_title` has been deprecated since version 3.14 and should not be used in newly-written code.

Use GNotification and GtkApplication to provide status notifications; there is no direct replacement for this function

Gets the title of this tray icon. See `gtk_status_icon_set_title()`.

Parameters

status_icon	a GtkStatusIcon

Returns

the title of the status icon

Since: 2.18

---

gtk_status_icon_set_name ()

```
void
gtk_status_icon_set_name (GtkStatusIcon *status_icon,
                          const gchar *name);
```

`gtk_status_icon_set_name` has been deprecated since version 3.14 and should not be used in newly-written code.

Use GNotification and GtkApplication to provide status notifications; there is no direct replacement for this function, as notifications are associated with a unique application identifier by GApplication

Sets the name of this tray icon. This should be a string identifying this icon. It is may be used for sorting the

icons in the tray and will not be shown to the user.

Parameters

status_icon	a GtkStatusIcon
name	the name

Since: 2.20

---

gtk_status_icon_set_visible ()

```
void
gtk_status_icon_set_visible (GtkStatusIcon *status_icon,
                             gboolean visible);
```

`gtk_status_icon_set_visible` has been deprecated since version 3.14 and should not be used in newly-written code.

Use GNotification and GtkApplication to provide status notifications; there is no direct replacement for this function, as notifications are managed by the platform

Shows or hides a status icon.

Parameters

status_icon	a GtkStatusIcon
visible	`TRUE` to show the status icon, `FALSE` to hide it

Since: 2.10

---

gtk_status_icon_get_visible ()

```
gboolean
gtk_status_icon_get_visible (GtkStatusIcon *status_icon);
```

`gtk_status_icon_get_visible` has been deprecated since version 3.14 and should not be used in newly-written code.

Use GNotification and GtkApplication to provide status notifications; there is no direct replacement for this function

Returns whether the status icon is visible or not. Note that being visible does not guarantee that the user can actually see the icon, see also `gtk_status_icon_is_embedded()`.

Parameters

status_icon	a GtkStatusIcon

Returns

TRUE if the status icon is visible

Since: 2.10

---

gtk_status_icon_is_embedded ()

```
gboolean
gtk_status_icon_is_embedded (GtkStatusIcon *status_icon);
```

gtk_status_icon_is_embedded has been deprecated since version 3.14 and should not be used in newly-written code.

Use GNotification and GtkApplication to provide status notifications; there is no direct replacement for this function

Returns whether the status icon is embedded in a notification area.

Parameters

status_icon	a GtkStatusIcon

Returns

TRUE if the status icon is embedded in a notification area.

Since: 2.10

---

gtk_status_icon_position_menu ()

```
void
gtk_status_icon_position_menu (GtkMenu *menu,
                               gint *x,
                               gint *y,
                               gboolean *push_in,
                               gpointer user_data);
```

gtk_status_icon_position_menu has been deprecated since version 3.14 and should not be used in newly-written code.

Use GNotification and GtkApplication to provide status notifications; notifications do not have menus, but can have buttons, and actions associated with each button

Menu positioning function to use with gtk_menu_popup() to position menu aligned to the status icon user_data .

Parameters

menu	the GtkMenu	
x	return location for the x position.	[inout]

y	return location for the y position.	[inout]
push_in	whether the first menu item should be offset (pushed in) to be aligned with the menu popup position (only useful for GtkOptionMenu).	[out]
user_data	the status icon to position the menu on.	[type GtkStatusIcon]

Since: 2.10

---

gtk_status_icon_get_geometry ()

```
gboolean
gtk_status_icon_get_geometry (GtkStatusIcon *status_icon,
                              GdkScreen **screen,
                              GdkRectangle *area,
                              GtkOrientation *orientation);
```

`gtk_status_icon_get_geometry` has been deprecated since version 3.14 and should not be used in newly-written code.

Use GNotification and GtkApplication to provide status notifications; there is no direct replacement for this function, as the platform is responsible for the presentation of notifications

Obtains information about the location of the status icon on screen. This information can be used to e.g. position popups like notification bubbles.

See `gtk_status_icon_position_menu()` for a more convenient alternative for positioning menus.

Note that some platforms do not allow GTK+ to provide this information, and even on platforms that do allow it, the information is not reliable unless the status icon is embedded in a notification area, see `gtk_status_icon_is_embedded()`.

Parameters

status_icon	a GtkStatusIcon	
screen	return location for the screen, or `NULL` if the information is not needed.	[out][transfer none][allow-none]
area	return location for the area occupied by the status icon, or `NULL`.	[out][allow-none]
orientation	return location for the orientation of the panel in which the status icon is embedded, or `NULL`. A panel at the top or bottom of the screen is horizontal, a panel at the left or right is vertical.	[out][allow-none]

Returns

`TRUE` if the location information has been filled in

Since: 2.10

gtk_status_icon_get_x11_window_id ()

```
guint32
gtk_status_icon_get_x11_window_id (GtkStatusIcon *status_icon);
```

`gtk_status_icon_get_x11_window_id` has been deprecated since version 3.14 and should not be used in newly-written code.

Use GNotification and GtkApplication to provide status notifications; there is no direct replacement for this function

This function is only useful on the X11/freedesktop.org platform.

It returns a window ID for the widget in the underlying status icon implementation. This is useful for the Galago notification service, which can send a window ID in the protocol in order for the server to position notification windows pointing to a status icon reliably.

This function is not intended for other use cases which are more likely to be met by one of the non-X11 specific methods, such as `gtk_status_icon_position_menu()`.

Parameters

status_icon	a GtkStatusIcon

Returns

An 32 bit unsigned integer identifier for the underlying X11 Window

Since: 2.14

Types and Values

struct GtkStatusIcon

```
struct GtkStatusIcon;
```

Property Details

The “embedded” property

```
  “embedded”                 gboolean
```

`TRUE` if the statusicon is embedded in a notification area.

Flags: Read

Default value: FALSE

Since: 2.12

The “file” property

“file” gchar *

Filename to load and display.

Flags: Write

Default value: NULL

The “gicon” property

“gicon” GIcon *

The GIcon displayed in the GtkStatusIcon. For themed icons, the image will be updated automatically if the theme changes.

Flags: Read / Write

Since: 2.14

The “has-tooltip” property

“has-tooltip” gboolean

Enables or disables the emission of “query-tooltip” on status_icon . A value of TRUE indicates that status_icon can have a tooltip, in this case the status icon will be queried using “query-tooltip” to determine whether it will provide a tooltip or not.

Note that setting this property to TRUE for the first time will change the event masks of the windows of this status icon to include leave-notify and motion-notify events. This will not be undone when the property is set to FALSE again.

Whether this property is respected is platform dependent. For plain text tooltips, use “tooltip-text” in preference.

Flags: Read / Write

Default value: FALSE

Since: 2.16

The “icon-name” property

“icon-name” gchar *

The name of the icon from the icon theme.

Flags: Read / Write

Default value: NULL

The “orientation” property

“orientation” GtkOrientation

The orientation of the tray in which the statusicon is embedded.

Flags: Read

Default value: GTK_ORIENTATION_HORIZONTAL

Since: 2.12

The “pixbuf” property

“pixbuf” GdkPixbuf *

A GdkPixbuf to display.

Flags: Read / Write

The “screen” property

“screen” GdkScreen *

The screen where this status icon will be displayed.

Flags: Read / Write

The “size” property

“size” gint

The size of the icon.

Flags: Read

Allowed values: >= 0

Default value: 0

The “stock” property

“stock” gchar *

Stock ID for a stock image to display.

`GtkStatusIcon:stock` has been deprecated since version 3.10 and should not be used in newly-written code.

Use “icon-name” instead.

Flags: Read / Write

Default value: NULL

The “storage-type” property

“storage-type” GtkImageType

The representation being used for image data.

Flags: Read

Default value: GTK_IMAGE_EMPTY

The “title” property

“title” gchar *

The title of this tray icon. This should be a short, human-readable, localized string describing the tray icon. It may be used by tools like screen readers to render the tray icon.

Flags: Read / Write

Default value: NULL

Since: 2.18

The “tooltip-markup” property

“tooltip-markup” gchar *

Sets the text of tooltip to be the given string, which is marked up with the Pango text markup language. Also see gtk_tooltip_set_markup().

This is a convenience property which will take care of getting the tooltip shown if the given string is not NULL. “has-tooltip” will automatically be set to TRUE and the default handler for the “query-tooltip” signal will take care of displaying the tooltip.

On some platforms, embedded markup will be ignored.

Flags: Read / Write

Default value: NULL

Since: 2.16

The “tooltip-text” property

“tooltip-text” gchar *

Sets the text of tooltip to be the given string.

Also see gtk_tooltip_set_text().

This is a convenience property which will take care of getting the tooltip shown if the given string is not NULL. “has-tooltip” will automatically be set to TRUE and the default handler for the “query-tooltip” signal will take

care of displaying the tooltip.

Note that some platforms have limitations on the length of tooltips that they allow on status icons, e.g. Windows only shows the first 64 characters.

Flags: Read / Write

Default value: NULL

Since: 2.16

The “visible” property

```
  “visible”                   gboolean
```

Whether the status icon is visible.

Flags: Read / Write

Default value: TRUE

Signal Details

The “activate” signal

```
void
user_function (GtkStatusIcon *status_icon,
               gpointer       user_data)
```

Gets emitted when the user activates the status icon. If and how status icons can activated is platform-dependent.

Unlike most G_SIGNAL_ACTION signals, this signal is meant to be used by applications and should be wrapped by language bindings.

Parameters

status_icon	the object which received the signal
user_data	user data set when the signal handler was connected.

Flags: Action

Since: 2.10

The “button-press-event” signal

```
gboolean
user_function (GtkStatusIcon *status_icon,
               GdkEvent      *event,
               gpointer       user_data)
```

The ::button-press-event signal will be emitted when a button (typically from a mouse) is pressed.

Whether this event is emitted is platform-dependent. Use the ::activate and ::popup-menu signals in preference.

Parameters

status_icon	the object which received the signal	
event	the GdkEventButton which triggered this signal.	[type Gdk.EventButton]
user_data	user data set when the signal handler was connected.	

Returns

`TRUE` to stop other handlers from being invoked for the event. `FALSE` to propagate the event further.

Flags: Run Last

Since: 2.14

The `"button-release-event"` signal

```
gboolean
user_function (GtkStatusIcon *status_icon,
               GdkEvent      *event,
               gpointer       user_data)
```

The ::button-release-event signal will be emitted when a button (typically from a mouse) is released.

Whether this event is emitted is platform-dependent. Use the ::activate and ::popup-menu signals in preference.

Parameters

status_icon	the object which received the signal	
event	the GdkEventButton which triggered this signal.	[type Gdk.EventButton]
user_data	user data set when the signal handler was connected.	

Returns

`TRUE` to stop other handlers from being invoked for the event. `FALSE` to propagate the event further.

Flags: Run Last

Since: 2.14

The “popup-menu” signal

```
void
user_function (GtkStatusIcon *status_icon,
               guint          button,
               guint          activate_time,
               gpointer       user_data)
```

Gets emitted when the user brings up the context menu of the status icon. Whether status icons can have context menus and how these are activated is platform-dependent.

The `button` and `activate_time` parameters should be passed as the last to arguments to `gtk_menu_popup()`.

Unlike most G_SIGNAL_ACTION signals, this signal is meant to be used by applications and should be wrapped by language bindings.

Parameters

status_icon	the object which received the signal
button	the button that was pressed, or 0 if the signal is not emitted in response to a button press event
activate_time	the timestamp of the event that triggered the signal emission
user_data	user data set when the signal handler was connected.

Flags: Action

Since: 2.10

The “query-tooltip” signal

```
gboolean
user_function (GtkStatusIcon *status_icon,
               gint           x,
               gint           y,
               gboolean       keyboard_mode,
               GtkTooltip    *tooltip,
               gpointer       user_data)
```

Emitted when the hover timeout has expired with the cursor hovering above `status_icon` ; or emitted when `status_icon` got focus in keyboard mode.

Using the given coordinates, the signal handler should determine whether a tooltip should be shown for `status_icon` . If this is the case `TRUE` should be returned, `FALSE` otherwise. Note that if `keyboard_mode` is `TRUE`, the values of `x` and `y` are undefined and should not be used.

The signal handler is free to manipulate `tooltip` with the therefore destined function calls.

Whether this signal is emitted is platform-dependent. For plain text tooltips, use “tooltip-text” in preference.

Parameters

status_icon	the object which received the signal

x	the x coordinate of the cursor position where the request has been emitted, relative to `status_icon`
y	the y coordinate of the cursor position where the request has been emitted, relative to `status_icon`
keyboard_mode	`TRUE` if the tooltip was trigged using the keyboard
tooltip	a GtkTooltip
user_data	user data set when the signal handler was connected.

Returns

`TRUE` if `tooltip` should be shown right now, `FALSE` otherwise.

Flags: Run Last

Since: 2.16

The `"scroll-event"` signal

```
gboolean
user_function (GtkStatusIcon *status_icon,
               GdkEvent      *event,
               gpointer       user_data)
```

The ::scroll-event signal is emitted when a button in the 4 to 7 range is pressed. Wheel mice are usually configured to generate button press events for buttons 4 and 5 when the wheel is turned.

Whether this event is emitted is platform-dependent.

Parameters

status_icon	the object which received the signal.
event	the GdkEventScroll which triggered this signal. [type Gdk.EventScroll]
user_data	user data set when the signal handler was connected.

Returns

`TRUE` to stop other handlers from being invoked for the event. `FALSE` to propagate the event further.

Flags: Run Last

Since: 2.16

The "size-changed" signal

```
gboolean
user_function (GtkStatusIcon *status_icon,
               gint           size,
               gpointer       user_data)
```

Gets emitted when the size available for the image changes, e.g. because the notification area got resized.

Parameters

status_icon	the object which received the signal
size	the new size
user_data	user data set when the signal handler was connected.

Returns

TRUE if the icon was updated for the new size. Otherwise, GTK+ will scale the icon as necessary.

Flags: Run Last

Since: 2.10

GtkThemingEngine

GtkThemingEngine — Theming renderers

Functions

void	gtk_theming_engine_get ()
GtkTextDirection	gtk_theming_engine_get_direction ()
GtkJunctionSides	gtk_theming_engine_get_junction_sides ()
const GtkWidgetPath *	gtk_theming_engine_get_path ()
void	gtk_theming_engine_get_property ()
GdkScreen *	gtk_theming_engine_get_screen ()
GtkStateFlags	gtk_theming_engine_get_state ()
void	gtk_theming_engine_get_style ()
void	gtk_theming_engine_get_style_property ()
void	gtk_theming_engine_get_style_valist ()
void	gtk_theming_engine_get_valist ()
void	gtk_theming_engine_get_color ()
void	gtk_theming_engine_get_background_color ()
void	gtk_theming_engine_get_border_color ()
void	gtk_theming_engine_get_border ()
void	gtk_theming_engine_get_padding ()
void	gtk_theming_engine_get_margin ()
const PangoFontDescription *	gtk_theming_engine_get_font ()
gboolean	gtk_theming_engine_has_class ()
gboolean	gtk_theming_engine_has_region ()
gboolean	gtk_theming_engine_lookup_color ()

gboolean	gtk_theming_engine_state_is_running ()
GtkThemingEngine *	gtk_theming_engine_load ()
void	gtk_theming_engine_register_property ()

Properties

gchar *	name	Read / Write / Construct Only

Types and Values

struct	GtkThemingEngineClass
struct	GtkThemingEngine

Object Hierarchy

```
GObject
└── GtkThemingEngine
```

Includes

```
#include <gtk/gtk.h>
```

Description

GtkThemingEngine was the object used for rendering themed content in GTK+ widgets. It used to allow overriding GTK+'s default implementation of rendering functions by allowing engines to be loaded as modules.

GtkThemingEngine has been deprecated in GTK+ 3.14 and will be ignored for rendering. The advancements in CSS theming are good enough to allow themers to achieve their goals without the need to modify source code.

Functions

gtk_theming_engine_get ()

```
void
gtk_theming_engine_get (GtkThemingEngine *engine,
                        GtkStateFlags state,
                        ...);
```

`gtk_theming_engine_get` has been deprecated since version 3.14 and should not be used in newly-written code.

Retrieves several style property values that apply to the currently rendered element.

Parameters

engine	a GtkThemingEngine
state	state to retrieve values for
...	property name /return value pairs,

followed by `NULL`

Since: 3.0

---

gtk_theming_engine_get_direction ()

```
GtkTextDirection
gtk_theming_engine_get_direction (GtkThemingEngine *engine);
```

`gtk_theming_engine_get_direction` has been deprecated since version 3.8 and should not be used in newly-written code.

Use `gtk_theming_engine_get_state()` and check for GTK_STATE_FLAG_DIR_LTR and GTK_STATE_FLAG_DIR_RTL instead.

Returns the widget direction used for rendering.

Parameters

engine	a GtkThemingEngine

Returns

the widget direction

Since: 3.0

---

gtk_theming_engine_get_junction_sides ()

```
GtkJunctionSides
gtk_theming_engine_get_junction_sides (GtkThemingEngine *engine);
```

`gtk_theming_engine_get_junction_sides` has been deprecated since version 3.14 and should not be used in newly-written code.

Returns the widget direction used for rendering.

Parameters

engine	a GtkThemingEngine

Returns

the widget direction

Since: 3.0

---

gtk_theming_engine_get_path ()

```
const GtkWidgetPath *
gtk_theming_engine_get_path (GtkThemingEngine *engine);
```

`gtk_theming_engine_get_path` has been deprecated since version 3.14 and should not be used in newly-written code.

Returns the widget path used for style matching.

Parameters

engine	a GtkThemingEngine	

Returns

A GtkWidgetPath.

[transfer none]

Since: 3.0

---

gtk_theming_engine_get_property ()

```
void
gtk_theming_engine_get_property (GtkThemingEngine *engine,
                                 const gchar *property,
                                 GtkStateFlags state,
                                 GValue *value);
```

`gtk_theming_engine_get_property` has been deprecated since version 3.14 and should not be used in newly-written code.

Gets a property value as retrieved from the style settings that apply to the currently rendered element.

Parameters

engine	a GtkThemingEngine	
property	the property name	
state	state to retrieve the value for	
value	return location for the property value, you must free this memory using `g_value_unset()` once you are done with it.	[out][transfer full]

Since: 3.0

---

gtk_theming_engine_get_screen ()

```
GdkScreen *
gtk_theming_engine_get_screen (GtkThemingEngine *engine);
```

`gtk_theming_engine_get_screen` has been deprecated since version 3.14 and should not be used in newly-

written code.

Returns the GdkScreen to which `engine` currently rendering to.

Parameters

engine	a GtkThemingEngine

Returns

a GdkScreen, or `NULL`.

[nullable][transfer none]

---

gtk_theming_engine_get_state ()

```
GtkStateFlags
gtk_theming_engine_get_state (GtkThemingEngine *engine);
```

`gtk_theming_engine_get_state` has been deprecated since version 3.14 and should not be used in newly-written code.

returns the state used when rendering.

Parameters

engine	a GtkThemingEngine

Returns

the state flags

Since: 3.0

---

gtk_theming_engine_get_style ()

```
void
gtk_theming_engine_get_style (GtkThemingEngine *engine,
                              ...);
```

`gtk_theming_engine_get_style` has been deprecated since version 3.14 and should not be used in newly-written code.

Retrieves several widget style properties from `engine` according to the currently rendered content's style.

Parameters

engine	a GtkThemingEngine
...	property name /return value pairs,

	followed by `NULL`	

Since: 3.0

---

gtk_theming_engine_get_style_property ()

```
void
gtk_theming_engine_get_style_property (GtkThemingEngine *engine,
                                       const gchar *property_name,
                                       GValue *value);
```

`gtk_theming_engine_get_style_property` has been deprecated since version 3.14 and should not be used in newly-written code.

Gets the value for a widget style property.

Parameters

engine	a GtkThemingEngine	
property_name	the name of the widget style property	
value	Return location for the property value, free with `g_value_unset()` after use.	[out]

Since: 3.0

---

gtk_theming_engine_get_style_valist ()

```
void
gtk_theming_engine_get_style_valist (GtkThemingEngine *engine,
                                     va_list args);
```

`gtk_theming_engine_get_style_valist` has been deprecated since version 3.14 and should not be used in newly-written code.

Retrieves several widget style properties from `engine` according to the currently rendered content's style.

Parameters

engine	a GtkThemingEngine	
args	va_list of property name/return location pairs, followed by `NULL`	

Since: 3.0

---

gtk_theming_engine_get_valist ()

```
void
gtk_theming_engine_get_valist (GtkThemingEngine *engine,
                               GtkStateFlags state,
```

```
                                va_list args);
```

`gtk_theming_engine_get_valist` has been deprecated since version 3.14 and should not be used in newly-written code.

Retrieves several style property values that apply to the currently rendered element.

Parameters

engine	a GtkThemingEngine	
state	state to retrieve values for	
args	va_list of property name/return location pairs, followed by `NULL`	

Since: 3.0

---

gtk_theming_engine_get_color ()

```
void
gtk_theming_engine_get_color (GtkThemingEngine *engine,
                              GtkStateFlags state,
                              GdkRGBA *color);
```

`gtk_theming_engine_get_color` has been deprecated since version 3.14 and should not be used in newly-written code.

Gets the foreground color for a given state.

Parameters

engine	a GtkThemingEngine	
state	state to retrieve the color for	
color	return value for the foreground color.	[out]

Since: 3.0

---

gtk_theming_engine_get_background_color ()

```
void
gtk_theming_engine_get_background_color
                              (GtkThemingEngine *engine,
                               GtkStateFlags state,
                               GdkRGBA *color);
```

`gtk_theming_engine_get_background_color` has been deprecated since version 3.14 and should not be used in newly-written code.

Gets the background color for a given state.

Parameters

engine	a GtkThemingEngine

state	state to retrieve the color for	
color	return value for the background color.	[out]

Since: 3.0

---

gtk_theming_engine_get_border_color ()

```
void
gtk_theming_engine_get_border_color (GtkThemingEngine *engine,
                                     GtkStateFlags state,
                                     GdkRGBA *color);
```

`gtk_theming_engine_get_border_color` has been deprecated since version 3.14 and should not be used in newly-written code.

Gets the border color for a given state.

Parameters

engine	a GtkThemingEngine	
state	state to retrieve the color for	
color	return value for the border color.	[out]

Since: 3.0

---

gtk_theming_engine_get_border ()

```
void
gtk_theming_engine_get_border (GtkThemingEngine *engine,
                               GtkStateFlags state,
                               GtkBorder *border);
```

`gtk_theming_engine_get_border` has been deprecated since version 3.14 and should not be used in newly-written code.

Gets the border for a given state as a GtkBorder.

Parameters

engine	a GtkThemingEngine	
state	state to retrieve the border for	
border	return value for the border settings.	[out]

Since: 3.0

---

gtk_theming_engine_get_padding ()

```
void
gtk_theming_engine_get_padding (GtkThemingEngine *engine,
                                GtkStateFlags state,
```

```
                                GtkBorder *padding);
```

`gtk_theming_engine_get_padding` has been deprecated since version 3.14 and should not be used in newly-written code.

Gets the padding for a given state as a GtkBorder.

Parameters

engine	a GtkThemingEngine	
state	state to retrieve the padding for	
padding	return value for the padding settings.	[out]

Since: 3.0

---

gtk_theming_engine_get_margin ()

```
void
gtk_theming_engine_get_margin (GtkThemingEngine *engine,
                               GtkStateFlags state,
                               GtkBorder *margin);
```

`gtk_theming_engine_get_margin` has been deprecated since version 3.14 and should not be used in newly-written code.

Gets the margin for a given state as a GtkBorder.

Parameters

engine	a GtkThemingEngine	
state	state to retrieve the border for	
margin	return value for the margin settings.	[out]

Since: 3.0

---

gtk_theming_engine_get_font ()

```
const PangoFontDescription *
gtk_theming_engine_get_font (GtkThemingEngine *engine,
                             GtkStateFlags state);
```

`gtk_theming_engine_get_font` has been deprecated since version 3.8 and should not be used in newly-written code.

Use `gtk_theming_engine_get()`

Returns the font description for a given state.

Parameters

engine	a GtkThemingEngine
state	state to retrieve the font for

Returns

the PangoFontDescription for the given state. This object is owned by GTK+ and should not be freed.

[transfer none]

Since: 3.0

---

gtk_theming_engine_has_class ()

```
gboolean
gtk_theming_engine_has_class (GtkThemingEngine *engine,
                              const gchar *style_class);
```

`gtk_theming_engine_has_class` has been deprecated since version 3.14 and should not be used in newly-written code.

Returns `TRUE` if the currently rendered contents have defined the given class name.

Parameters

engine	a GtkThemingEngine
style_class	class name to look up

Returns

`TRUE` if `engine` has `class_name` defined

Since: 3.0

---

gtk_theming_engine_has_region ()

```
gboolean
gtk_theming_engine_has_region (GtkThemingEngine *engine,
                               const gchar *style_region,
                               GtkRegionFlags *flags);
```

`gtk_theming_engine_has_region` has been deprecated since version 3.14 and should not be used in newly-written code.

Returns `TRUE` if the currently rendered contents have the region defined. If `flags_return` is not `NULL`, it is set to the flags affecting the region.

Parameters

engine	a GtkThemingEngine	
style_region	a region name	
flags	return location for region flags.	[out][allow-none]

Returns

TRUE if region is defined

Since: 3.0

---

gtk_theming_engine_lookup_color ()

```
gboolean
gtk_theming_engine_lookup_color (GtkThemingEngine *engine,
                                 const gchar *color_name,
                                 GdkRGBA *color);
```

gtk_theming_engine_lookup_color has been deprecated since version 3.14 and should not be used in newly-written code.

Looks up and resolves a color name in the current style's color map.

Parameters

engine	a GtkThemingEngine	
color_name	color name to lookup	
color	Return location for the looked up color.	[out]

Returns

TRUE if color_name was found and resolved, FALSE otherwise

Since: 3.0

---

gtk_theming_engine_state_is_running ()

```
gboolean
gtk_theming_engine_state_is_running (GtkThemingEngine *engine,
                                     GtkStateType state,
                                     gdouble *progress);
```

gtk_theming_engine_state_is_running has been deprecated since version 3.6 and should not be used in newly-written code.

Always returns FALSE

Returns TRUE if there is a transition animation running for the current region (see gtk_style_context_push_animatable_region()).

If progress is not NULL, the animation progress will be returned there, 0.0 means the state is closest to being FALSE, while 1.0 means it's closest to being TRUE. This means transition animations will run from 0 to 1 when state is being set to TRUE and from 1 to 0 when it's being set to FALSE.

Parameters

engine	a GtkThemingEngine	
state	a widget state	
progress	return location for the transition progress.	[out]

Returns

`TRUE` if there is a running transition animation for `state` .

Since: 3.0

---

gtk_theming_engine_load ()

```
GtkThemingEngine *
gtk_theming_engine_load (const gchar *name);
```

`gtk_theming_engine_load` has been deprecated since version 3.14 and should not be used in newly-written code.

Loads and initializes a theming engine module from the standard directories.

Parameters

name	Theme engine name to load

Returns

A theming engine, or `NULL` if the engine `name` doesn't exist.

[nullable][transfer none]

---

gtk_theming_engine_register_property ()

```
void
gtk_theming_engine_register_property (const gchar *name_space,
                                      GtkStylePropertyParser parse_func,
                                      GParamSpec *pspec);
```

`gtk_theming_engine_register_property` has been deprecated since version 3.8 and should not be used in newly-written code.

Code should use the default properties provided by CSS.

Registers a property so it can be used in the CSS file format, on the CSS file the property will look like "-${`name_space` }-${property_name}". being ${property_name} the given to `pspec` . `name_space` will usually be the theme engine name.

For any type a `parse_func` may be provided, being this function used for turning any property value (between ":" and ";") in CSS to the GValue needed. For basic types there is already builtin parsing support, so `NULL` may be provided for these cases.

Engines must ensure property registration happens exactly once, usually GTK+ deals with theming engines as singletons, so this should be guaranteed to happen once, but bear this in mind when creating GtkThemeEngines yourself.

In order to make use of the custom registered properties in the CSS file, make sure the engine is loaded first by specifying the engine property, either in a previous rule or within the same one.

```
1                                                   * {
2                                                       engine: someengine;
3                                                       -SomeEngine-custom-property: 2;
4                                                   }
```

[skip]

Parameters

name_space	namespace for the property name	
parse_func	parsing function to use, or `NULL`.	[nullable]
pspec	the GParamSpec for the new property	

Since: 3.0

Types and Values

struct GtkThemingEngineClass

```
struct GtkThemingEngineClass {
  GObjectClass parent_class;


  void (* render_line) (GtkThemingEngine *engine,
                        cairo_t          *cr,
                        gdouble           x0,
                        gdouble           y0,
                        gdouble           x1,
                        gdouble           y1);
  void (* render_background) (GtkThemingEngine *engine,
                              cairo_t          *cr,
                              gdouble           x,
                              gdouble           y,
                              gdouble           width,
                              gdouble           height);
  void (* render_frame) (GtkThemingEngine *engine,
                         cairo_t          *cr,
                         gdouble           x,
                         gdouble           y,
                         gdouble           width,
                         gdouble           height);
  void (* render_frame_gap) (GtkThemingEngine *engine,
                             cairo_t          *cr,
                             gdouble           x,
                             gdouble           y,
                             gdouble           width,
                             gdouble           height,
                             GtkPositionType   gap_side,
                             gdouble           xy0_gap,
```

```
                            gdouble           xy1_gap);
  void (* render_extension) (GtkThemingEngine *engine,
                             cairo_t          *cr,
                             gdouble           x,
                             gdouble           y,
                             gdouble           width,
                             gdouble           height,
                             GtkPositionType   gap_side);
  void (* render_check) (GtkThemingEngine *engine,
                         cairo_t          *cr,
                         gdouble           x,
                         gdouble           y,
                         gdouble           width,
                         gdouble           height);
  void (* render_option) (GtkThemingEngine *engine,
                          cairo_t          *cr,
                          gdouble           x,
                          gdouble           y,
                          gdouble           width,
                          gdouble           height);
  void (* render_arrow) (GtkThemingEngine *engine,
                         cairo_t          *cr,
                         gdouble           angle,
                         gdouble           x,
                         gdouble           y,
                         gdouble           size);
  void (* render_expander) (GtkThemingEngine *engine,
                            cairo_t          *cr,
                            gdouble           x,
                            gdouble           y,
                            gdouble           width,
                            gdouble           height);
  void (* render_focus) (GtkThemingEngine *engine,
                         cairo_t          *cr,
                         gdouble           x,
                         gdouble           y,
                         gdouble           width,
                         gdouble           height);
  void (* render_layout) (GtkThemingEngine *engine,
                          cairo_t          *cr,
                          gdouble           x,
                          gdouble           y,
                          PangoLayout      *layout);
  void (* render_slider) (GtkThemingEngine *engine,
                          cairo_t          *cr,
                          gdouble           x,
                          gdouble           y,
                          gdouble           width,
                          gdouble           height,
                          GtkOrientation    orientation);
  void (* render_handle)    (GtkThemingEngine *engine,
                             cairo_t          *cr,
                             gdouble           x,
                             gdouble           y,
                             gdouble           width,
                             gdouble           height);
  void (* render_activity) (GtkThemingEngine *engine,
                            cairo_t          *cr,
                            gdouble           x,
                            gdouble           y,
                            gdouble           width,
                            gdouble           height);
```

```
  GdkPixbuf * (* render_icon_pixbuf) (GtkThemingEngine    *engine,
                                      const GtkIconSource *source,
                                      GtkIconSize          size);
  void (* render_icon) (GtkThemingEngine *engine,
                        cairo_t          *cr,
                  GdkPixbuf        *pixbuf,
                        gdouble           x,
                        gdouble           y);
  void (* render_icon_surface) (GtkThemingEngine *engine,
                        cairo_t          *cr,
                        cairo_surface_t  *surface,
                        gdouble           x,
                        gdouble           y);
};
```

Base class for theming engines.

Members

`render_line` ()	Renders a line between two points.
`render_background` ()	Renders the background area of a widget region.
`render_frame` ()	Renders the frame around a widget area.
`render_frame_gap` ()	Renders the frame around a widget area with a gap in it.
`render_extension` ()	Renders a extension to a box, usually a notebook tab.
`render_check` ()	Renders a checkmark, as in GtkCheckButton.
`render_option` ()	Renders an option, as in GtkRadioButton.
`render_arrow` ()	Renders an arrow pointing to a certain direction.
`render_expander` ()	Renders an element what will expose/expand part of the UI, as in GtkExpander.
`render_focus` ()	Renders the focus indicator.
`render_layout` ()	Renders a PangoLayout
`render_slider` ()	Renders a slider control, as in GtkScale.
`render_handle` ()	Renders a handle to drag UI elements, as in GtkPaned.
`render_activity` ()	Renders an area displaying activity, such as in GtkSpinner, or GtkProgressBar.
`render_icon_pixbuf` ()	Renders an icon as a GdkPixbuf.
`render_icon` ()	Renders an icon given as a GdkPixbuf.
`render_icon_surface` ()	Renders an icon given as a cairo_surface_t.

struct GtkThemingEngine

```
struct GtkThemingEngine;
```

Property Details

The “name” property

```
  “name”                     gchar *
```

The theming engine name, this name will be used when registering custom properties, for a theming engine named "Clearlooks" registering a "glossy" custom property, it could be referenced in the CSS file as

```
1                                                -Clearlooks-glossy: true;
```

Flags: Read / Write / Construct Only

Default value: NULL

Since: 3.0

See Also

GtkStyleContext

GtkAlignment

GtkAlignment — A widget which controls the alignment and size of its child

Functions

GtkWidget *	gtk_alignment_new ()
void	gtk_alignment_set ()
void	gtk_alignment_get_padding ()
void	gtk_alignment_set_padding ()

Properties

guint	bottom-padding	Read / Write
guint	left-padding	Read / Write
guint	right-padding	Read / Write
guint	top-padding	Read / Write
gfloat	xalign	Read / Write
gfloat	xscale	Read / Write
gfloat	yalign	Read / Write
gfloat	yscale	Read / Write

Types and Values

struct	GtkAlignment
struct	GtkAlignmentClass

Object Hierarchy

```
GObject
└── GInitiallyUnowned
    └── GtkWidget
        └── GtkContainer
            └── GtkBin
                └── GtkAlignment
```

Implemented Interfaces

GtkAlignment implements AtkImplementorIface and GtkBuildable.

Includes

```
#include <gtk/gtk.h>
```

Description

The GtkAlignment widget controls the alignment and size of its child widget. It has four settings: xscale, yscale, xalign, and yalign.

The scale settings are used to specify how much the child widget should expand to fill the space allocated to the GtkAlignment. The values can range from 0 (meaning the child doesn't expand at all) to 1 (meaning the child expands to fill all of the available space).

The align settings are used to place the child widget within the available area. The values range from 0 (top or left) to 1 (bottom or right). Of course, if the scale settings are both set to 1, the alignment settings have no effect.

GtkAlignment has been deprecated in 3.14 and should not be used in newly-written code. The desired effect can be achieved by using the "halign", "valign" and "margin" properties on the child widget.

Functions

gtk_alignment_new ()

```
GtkWidget *
gtk_alignment_new (gfloat xalign,
                   gfloat yalign,
                   gfloat xscale,
                   gfloat yscale);
```

`gtk_alignment_new` has been deprecated since version 3.14 and should not be used in newly-written code.

Use GtkWidget alignment and margin properties

Creates a new GtkAlignment.

Parameters

xalign	the horizontal alignment of the child widget, from 0 (left) to 1 (right).
yalign	the vertical alignment of the child widget, from 0 (top) to 1 (bottom).
xscale	the amount that the child widget expands horizontally to fill up unused space, from 0 to 1. A value of 0 indicates that the child widget should never expand. A value of 1 indicates that the child widget will expand to fill all of the space allocated for the GtkAlignment.
yscale	the amount that the child widget expands vertically to fill up unused space, from 0 to 1. The values are similar to `xscale` .

Returns

the new GtkAlignment

---

gtk_alignment_set ()

```
void
gtk_alignment_set (GtkAlignment *alignment,
                   gfloat xalign,
                   gfloat yalign,
                   gfloat xscale,
                   gfloat yscale);
```

`gtk_alignment_set` has been deprecated since version 3.14 and should not be used in newly-written code.

Use GtkWidget alignment and margin properties

Sets the GtkAlignment values.

Parameters

alignment	a GtkAlignment.
xalign	the horizontal alignment of the child widget, from 0 (left) to 1 (right).
yalign	the vertical alignment of the child widget, from 0 (top) to 1 (bottom).
xscale	the amount that the child widget expands horizontally to fill up unused space, from 0 to 1. A value

	of 0 indicates that the child widget should never expand. A value of 1 indicates that the child widget will expand to fill all of the space allocated for the GtkAlignment.
yscale	the amount that the child widget expands vertically to fill up unused space, from 0 to 1. The values are similar to `xscale` .

---

gtk_alignment_get_padding ()

```
void
gtk_alignment_get_padding (GtkAlignment *alignment,
                           guint *padding_top,
                           guint *padding_bottom,
                           guint *padding_left,
                           guint *padding_right);
```

`gtk_alignment_get_padding` has been deprecated since version 3.14 and should not be used in newly-written code.

Use GtkWidget alignment and margin properties

Gets the padding on the different sides of the widget. See `gtk_alignment_set_padding()`.

Parameters

alignment	a GtkAlignment	
padding_top	location to store the padding for the top of the widget, or `NULL`.	[out][allow-none]
padding_bottom	location to store the padding for the bottom of the widget, or `NULL`.	[out][allow-none]
padding_left	location to store the padding for the left of the widget, or `NULL`.	[out][allow-none]
padding_right	location to store the padding for the right of the widget, or `NULL`.	[out][allow-none]

Since: 2.4

---

gtk_alignment_set_padding ()

```
void
gtk_alignment_set_padding (GtkAlignment *alignment,
                           guint padding_top,
                           guint padding_bottom,
                           guint padding_left,
                           guint padding_right);
```

`gtk_alignment_set_padding` has been deprecated since version 3.14 and should not be used in newly-written code.

Use GtkWidget alignment and margin properties

Sets the padding on the different sides of the widget. The padding adds blank space to the sides of the widget. For instance, this can be used to indent the child widget towards the right by adding padding on the left.

Parameters

alignment	a GtkAlignment
padding_top	the padding at the top of the widget
padding_bottom	the padding at the bottom of the widget
padding_left	the padding at the left of the widget
padding_right	the padding at the right of the widget.

Since: 2.4

Types and Values

struct GtkAlignment

```
struct GtkAlignment;
```

---

struct GtkAlignmentClass

```
struct GtkAlignmentClass {
  GtkBinClass parent_class;
};
```

Members

Property Details

The `"bottom-padding"` property

```
  "bottom-padding"           guint
```

The padding to insert at the bottom of the widget.

`GtkAlignment:bottom-padding` has been deprecated since version 3.14 and should not be used in newly-written code.

Use `gtk_widget_set_margin_bottom()` instead

Flags: Read / Write

Allowed values: <= G_MAXINT

Default value: 0

Since: 2.4

The “left-padding” property

“left-padding” guint

The padding to insert at the left of the widget.

GtkAlignment:left-padding has been deprecated since version 3.14 and should not be used in newly-written code.

Use gtk_widget_set_margin_start() instead

Flags: Read / Write

Allowed values: <= G_MAXINT

Default value: 0

Since: 2.4

The “right-padding” property

“right-padding” guint

The padding to insert at the right of the widget.

GtkAlignment:right-padding has been deprecated since version 3.14 and should not be used in newly-written code.

Use gtk_widget_set_margin_end() instead

Flags: Read / Write

Allowed values: <= G_MAXINT

Default value: 0

Since: 2.4

The “top-padding” property

“top-padding” guint

The padding to insert at the top of the widget.

GtkAlignment:top-padding has been deprecated since version 3.14 and should not be used in newly-written code.

Use gtk_widget_set_margin_top() instead

Flags: Read / Write

Allowed values: <= G_MAXINT

Default value: 0

Since: 2.4

The “xalign” property

“xalign” gfloat

Horizontal position of child in available space. A value of 0.0 will flush the child left (or right, in RTL locales); a value of 1.0 will flush the child right (or left, in RTL locales).

GtkAlignment:xalign has been deprecated since version 3.14 and should not be used in newly-written code.

Use gtk_widget_set_halign() on the child instead

Flags: Read / Write

Allowed values: [0,1]

Default value: 0.5

The “xscale” property

“xscale” gfloat

If available horizontal space is bigger than needed, how much of it to use for the child. A value of 0.0 means none; a value of 1.0 means all.

GtkAlignment:xscale has been deprecated since version 3.14 and should not be used in newly-written code.

Use gtk_widget_set_hexpand() on the child instead

Flags: Read / Write

Allowed values: [0,1]

Default value: 1

The “yalign” property

“yalign” gfloat

Vertical position of child in available space. A value of 0.0 will flush the child to the top; a value of 1.0 will flush the child to the bottom.

GtkAlignment:yalign has been deprecated since version 3.14 and should not be used in newly-written code.

Use gtk_widget_set_valign() on the child instead

Flags: Read / Write

Allowed values: [0,1]

Default value: 0.5

The “yscale” property

“yscale” gfloat

If available vertical space is bigger than needed, how much of it to use for the child. A value of 0.0 means none; a value of 1.0 means all.

GtkAlignment:yscale has been deprecated since version 3.14 and should not be used in newly-written code.

Use gtk_widget_set_vexpand() on the child instead

Flags: Read / Write

Allowed values: [0,1]

Default value: 1
