Part III. GTK+ Core Reference: GTK+ 3 Reference Manual

Table of Contents

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

Main loop and Events

Main loop and Events — Library initialization, main event loop, and events

Functions

void	gtk_disable_setlocale ()
PangoLanguage *	gtk_get_default_language ()
GtkTextDirection	gtk_get_locale_direction ()
gboolean	gtk_parse_args ()
void	gtk_init ()
gboolean	gtk_init_check ()
gboolean	gtk_init_with_args ()
GOptionGroup *	gtk_get_option_group ()
gboolean	gtk_events_pending ()
void	gtk_main ()
guint	gtk_main_level ()
void	gtk_main_quit ()
gboolean	gtk_main_iteration ()
gboolean	gtk_main_iteration_do ()
void	gtk_main_do_event ()
void	(*GtkModuleInitFunc) ()
void	(*GtkModuleDisplayInitFunc) ()
gboolean	gtk_true ()
gboolean	gtk_false ()
void	gtk_grab_add ()
GtkWidget *	gtk_grab_get_current ()
void	gtk_grab_remove ()
void	gtk_device_grab_add ()
void	gtk_device_grab_remove ()
guint	gtk_key_snooper_install ()
gint	(*GtkKeySnoopFunc) ()
void	gtk_key_snooper_remove ()
GdkEvent *	gtk_get_current_event ()
guint32	gtk_get_current_event_time ()
gboolean	gtk_get_current_event_state ()
GdkDevice *	gtk_get_current_event_device ()
GtkWidget *	gtk_get_event_widget ()
void	gtk_propagate_event ()

Types and Values

#define	GTK_PRIORITY_RESIZE

Includes

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

Description

Before using GTK+, you need to initialize it; initialization connects to the window system display, and parses some standard command line arguments. The `gtk_init()` macro initializes GTK+. `gtk_init()` exits the application if errors occur; to avoid this, use `gtk_init_check()`. `gtk_init_check()` allows you to recover from a failed GTK+ initialization - you might start up your application in text mode instead.

Like all GUI toolkits, GTK+ uses an event-driven programming model. When the user is doing nothing, GTK+ sits in the “main loop” and waits for input. If the user performs some action - say, a mouse click - then the main loop “wakes up” and delivers an event to GTK+. GTK+ forwards the event to one or more widgets.

When widgets receive an event, they frequently emit one or more “signals”. Signals notify your program that "something interesting happened" by invoking functions you’ve connected to the signal with `g_signal_connect()`. Functions connected to a signal are often termed “callbacks”.

When your callbacks are invoked, you would typically take some action - for example, when an Open button is clicked you might display a GtkFileChooserDialog. After a callback finishes, GTK+ will return to the main loop and await more user input.

Typical `main()` function for a GTK+ application

```
1  int
2  main (int argc, char **argv)
3  {
4   GtkWidget *mainwin;
5    // Initialize i18n support with bindtextdomain(), etc.
6
7    // ...
8
9    // Initialize the widget set
10   gtk_init (&argc, &argv);
11
12   // Create the main window
13   mainwin = gtk_window_new (GTK_WINDOW_TOPLEVEL);
14
15   // Set up our GUI elements
16
17   // ...
18
19   // Show the application window
20   gtk_widget_show_all (mainwin);
21
22   // Enter the main event loop, and wait for user interaction
23   gtk_main ();
24
25   // The user lost interest
26   return 0;
27 }
```

It’s OK to use the GLib main loop directly instead of `gtk_main()`, though it involves slightly more typing. See GMainLoop in the GLib documentation.

Functions

gtk_disable_setlocale ()

```
void
gtk_disable_setlocale (void);
```

Prevents `gtk_init()`, `gtk_init_check()`, `gtk_init_with_args()` and `gtk_parse_args()` from automatically calling `setlocale (LC_ALL, "")`. You would want to use this function if you wanted to set the locale for your program to something other than the user's locale, or if you wanted to set different values for different locale categories.

Most programs should not need to call this function.

---

gtk_get_default_language ()

```
PangoLanguage *
gtk_get_default_language (void);
```

Returns the PangoLanguage for the default language currently in effect. (Note that this can change over the life of an application.) The default language is derived from the current locale. It determines, for example, whether GTK+ uses the right-to-left or left-to-right text direction.

This function is equivalent to `pango_language_get_default()`. See that function for details.

Returns

the default language as a PangoLanguage, must not be freed.

[transfer none]

---

gtk_get_locale_direction ()

```
GtkTextDirection
gtk_get_locale_direction (void);
```

Get the direction of the current locale. This is the expected reading direction for text and UI.

This function depends on the current locale being set with `setlocale()` and will default to setting the `GTK_TEXT_DIR_LTR` direction otherwise. `GTK_TEXT_DIR_NONE` will never be returned.

GTK+ sets the default text direction according to the locale during `gtk_init()`, and you should normally use `gtk_widget_get_direction()` or `gtk_widget_get_default_direction()` to obtain the current direcion.

This function is only needed rare cases when the locale is changed after GTK+ has already been initialized. In this case, you can use it to update the default text direction as follows:

```
1 setlocale (LC_ALL, new_locale);
2 direction = gtk_get_locale_direction ();
3 gtk_widget_set_default_direction (direction);
```

Returns

the GtkTextDirection of the current locale

Since: 3.12

---

gtk_parse_args ()

```
gboolean
gtk_parse_args (int *argc,
                char ***argv);
```

Parses command line arguments, and initializes global attributes of GTK+, but does not actually open a connection to a display. (See `gdk_display_open()`, `gdk_get_display_arg_name()`)

Any arguments used by GTK+ or GDK are removed from the array and `argc` and `argv` are updated accordingly.

There is no need to call this function explicitly if you are using `gtk_init()`, or `gtk_init_check()`.

Note that many aspects of GTK+ require a display connection to function, so this way of initializing GTK+ is really only useful for specialized use cases.

Parameters

argc	a pointer to the number of command line arguments.	[inout]
argv	a pointer to the array of command line arguments.	[array length=argc][inout]

Returns

`TRUE` if initialization succeeded, otherwise `FALSE`

---

gtk_init ()

```
void
gtk_init (int *argc,
          char ***argv);
```

Call this function before using any other GTK+ functions in your GUI applications. It will initialize everything needed to operate the toolkit and parses some standard command line options.

Although you are expected to pass the `argc` , `argv` parameters from `main()` to this function, it is possible to pass `NULL` if `argv` is not available or commandline handling is not required.

`argc` and `argv` are adjusted accordingly so your own code will never see those standard arguments.

Note that there are some alternative ways to initialize GTK+: if you are calling `gtk_parse_args()`, `gtk_init_check()`, `gtk_init_with_args()` or `g_option_context_parse()` with the option group returned by `gtk_get_option_group()`, you don’t have to call `gtk_init()`.

And if you are using GtkApplication, you don't have to call any of the initialization functions either; the “startup” handler does it for you.

This function will terminate your program if it was unable to initialize the windowing system for some reason. If you want your program to fall back to a textual interface you want to call `gtk_init_check()` instead.

Since 2.18, GTK+ calls `signal (SIGPIPE, SIG_IGN)` during initialization, to ignore SIGPIPE signals, since these are almost never wanted in graphical applications. If you do need to handle SIGPIPE for some reason, reset the handler after `gtk_init()`, but notice that other libraries (e.g. libdbus or gvfs) might do similar things.

Parameters

argc	Address of the `argc` parameter of your `main()` function (or 0 if `argv` is `NULL`). This will be changed if any arguments were handled.	[inout]
argv	Address of the `argv` parameter of `main()`, or `NULL`. Any options understood by GTK+ are stripped before return.	[array length=argc][inout][allow-none]

gtk_init_check ()

```
gboolean
gtk_init_check (int *argc,
                char ***argv);
```

This function does the same work as gtk_init() with only a single change: It does not terminate the program if the commandline arguments couldn't be parsed or the windowing system can't be initialized. Instead it returns FALSE on failure.

This way the application can fall back to some other means of communication with the user - for example a curses or command line interface.

Parameters

argc	Address of the argc parameter of your main() function (or 0 if argv is NULL). This will be changed if any arguments were handled.	[inout]
argv	Address of the argv parameter of main(), or NULL. Any options understood by GTK+ are stripped before return.	[array length=argc][inout][allow-none]

Returns

TRUE if the commandline arguments (if any) were valid and the windowing system has been successfully initialized, FALSE otherwise

---

gtk_init_with_args ()

```
gboolean
gtk_init_with_args (gint *argc,
                    gchar ***argv,
                    const gchar *parameter_string,
                    const GOptionEntry *entries,
                    const gchar *translation_domain,
                    GError **error);
```

This function does the same work as `gtk_init_check()`. Additionally, it allows you to add your own commandline options, and it automatically generates nicely formatted `--help` output. Note that your program will be terminated after writing out the help output.

Parameters

argc	Address of the `argc` parameter of your `main()` function (or 0 if `argv` is `NULL`). This will be changed if any arguments were handled.	[inout]
argv	Address of the `argv` parameter of `main()`, or `NULL`. Any options understood by GTK+ are stripped before return.	[array length=argc][inout][allow-none]
parameter_string	a string which is displayed in the first line of `--help` output, after `programname [OPTION...]`.	[allow-none]
entries	a `NULL`-terminated array of GOptionEntrys describing the options of your program.	[array zero-terminated=1]
translation_domain	a translation domain to use for translating the `--help` output for the options in `entries` and the `parameter_string` with `gettext()`, or `NULL`.	[nullable]
error	a return location for errors	

Returns

`TRUE` if the commandline arguments (if any) were valid and if the windowing system has been successfully initialized, `FALSE` otherwise

Since: 2.6

---

gtk_get_option_group ()

```
GOptionGroup *
gtk_get_option_group (gboolean open_default_display);
```

Returns a GOptionGroup for the commandline arguments recognized by GTK+ and GDK.

You should add this group to your GOptionContext with `g_option_context_add_group()`, if you are using `g_option_context_parse()` to parse your commandline arguments.

Parameters

open_default_display	whether to open the default display when parsing the commandline arguments

Returns

a GOptionGroup for the commandline arguments recognized by GTK+.

[transfer full]

Since: 2.6

---

gtk_events_pending ()

```
gboolean
gtk_events_pending (void);
```

Checks if any events are pending.

This can be used to update the UI and invoke timeouts etc. while doing some time intensive computation.

Updating the UI during a long computation

```
1  // computation going on...
2
3  while (gtk_events_pending ())
4    gtk_main_iteration ();
5
6  // ...computation continued
```

Returns

`TRUE` if any events are pending, `FALSE` otherwise

---

gtk_main ()

```
void
gtk_main (void);
```

Runs the main loop until gtk_main_quit() is called.

You can nest calls to gtk_main(). In that case gtk_main_quit() will make the innermost invocation of the main loop return.

---

gtk_main_level ()

```
guint
gtk_main_level (void);
```

Asks for the current nesting level of the main loop.

Returns

the nesting level of the current invocation of the main loop

---

gtk_main_quit ()

```
void
gtk_main_quit (void);
```

Makes the innermost invocation of the main loop return when it regains control.

---

gtk_main_iteration ()

```
gboolean
gtk_main_iteration (void);
```

Runs a single iteration of the mainloop.

If no events are waiting to be processed GTK+ will block until the next event is noticed. If you don't want to block look at gtk_main_iteration_do() or check if any events are pending with gtk_events_pending() first.

Returns

TRUE if gtk_main_quit() has been called for the innermost mainloop

---

gtk_main_iteration_do ()

```
gboolean
gtk_main_iteration_do (gboolean blocking);
```

Runs a single iteration of the mainloop. If no events are available either return or block depending on the value of `blocking`.

Parameters

blocking	`TRUE` if you want GTK+ to block if no events are pending

Returns

`TRUE` if `gtk_main_quit()` has been called for the innermost mainloop

---

gtk_main_do_event ()

```
void
gtk_main_do_event (GdkEvent *event);
```

Processes a single GDK event.

This is public only to allow filtering of events between GDK and GTK+. You will not usually need to call this function directly.

While you should not call this function directly, you might want to know how exactly events are handled. So here is what this function does with the event:

1. Compress enter/leave notify events. If the event passed build an enter/leave pair together with the next event (peeked from GDK), both events are thrown away. This is to avoid a backlog of (de-)highlighting widgets crossed by the pointer.
2. Find the widget which got the event. If the widget can’t be determined the event is thrown away unless it belongs to a INCR transaction.
3. Then the event is pushed onto a stack so you can query the currently handled event with `gtk_get_current_event()`.
4. The event is sent to a widget. If a grab is active all events for widgets that are not in the contained in the grab widget are sent to the latter with a few exceptions:
   - Deletion and destruction events are still sent to the event widget for obvious reasons.
   - Events which directly relate to the visual representation of the event widget.
   - Leave events are delivered to the event widget if there was an enter event delivered to it before without the paired leave event.
   - Drag events are not redirected because it is unclear what the semantics of that would be. Another point of interest might be that all key events are first passed through the key snooper functions if there are any. Read the description of `gtk_key_snooper_install()` if you need this feature.
5. After finishing the delivery the event is popped from the event stack.

Parameters

event	An event to process (normally passed by GDK)

GtkModuleInitFunc ()

```
void
(*GtkModuleInitFunc) (gint *argc,
                      gchar ***argv);
```

Each GTK+ module must have a function `gtk_module_init()` with this prototype. This function is called after loading the module.

Parameters

argc	GTK+ always passes `NULL` for this argument.	[allow-none]
argv	GTK+ always passes `NULL` for this argument.	[allow-none][array length=argc]

GtkModuleDisplayInitFunc ()

```
void
(*GtkModuleDisplayInitFunc) (GdkDisplay *display);
```

A multihead-aware GTK+ module may have a `gtk_module_display_init()` function with this prototype. GTK+ calls this function for each opened display.

Parameters

display	an open GdkDisplay

Since: 2.2

---

gtk_true ()

```
gboolean
gtk_true (void);
```

All this function does it to return `TRUE`.

This can be useful for example if you want to inhibit the deletion of a window. Of course you should not do this as the user expects a reaction from clicking the close icon of the window...

A persistent window

```
1  #include <gtk/gtk.h>
2
3  int
4  main (int argc, char **argv)
5  {
6    GtkWidget *win, *but;
7    const char *text = "Close yourself. I mean it!";
8
9    gtk_init (&argc, &argv);
10
11   win = gtk_window_new (GTK_WINDOW_TOPLEVEL);
12   g_signal_connect (win,
13                     "delete-event",
14                     G_CALLBACK (gtk_true),
15                     NULL);
16   g_signal_connect (win, "destroy",
17                     G_CALLBACK (gtk_main_quit),
18                     NULL);
19
20   but = gtk_button_new_with_label (text);
21   g_signal_connect_swapped (but, "clicked",
22                             G_CALLBACK (gtk_object_destroy),
23                             win);
24   gtk_container_add (GTK_CONTAINER (win), but);
25
26   gtk_widget_show_all (win);
27
28   gtk_main ();
29
30   return 0;
31 }
```

Returns

`TRUE`

---

gtk_false ()

```
gboolean
gtk_false (void);
```

Analogical to `gtk_true()`, this function does nothing but always returns `FALSE`.

Returns

`FALSE`

---

gtk_grab_add ()

```
void
gtk_grab_add (GtkWidget *widget);
```

Makes `widget` the current grabbed widget.

This means that interaction with other widgets in the same application is blocked and mouse as well as keyboard events are delivered to this widget.

If `widget` is not sensitive, it is not set as the current grabbed widget and this function does nothing.

[method]

Parameters

widget	The widget that grabs keyboard and pointer events

---

gtk_grab_get_current ()

```
GtkWidget *
gtk_grab_get_current (void);
```

Queries the current grab of the default window group.

Returns

The widget which currently has the grab or `NULL` if no grab is active.

[transfer none][nullable]

---

gtk_grab_remove ()

```
void
gtk_grab_remove (GtkWidget *widget);
```

Removes the grab from the given widget.

You have to pair calls to `gtk_grab_add()` and `gtk_grab_remove()`.

If `widget` does not have the grab, this function does nothing.

[method]

Parameters

widget	The widget which gives up the grab

---

gtk_device_grab_add ()

```
void
gtk_device_grab_add (GtkWidget *widget,
                     GdkDevice *device,
                     gboolean block_others);
```

Adds a GTK+ grab on `device` , so all the events on `device` and its associated pointer or keyboard (if any) are delivered to `widget` . If the `block_others` parameter is `TRUE`, any other devices will be unable to interact with `widget` during the grab.

Parameters

widget	a GtkWidget
device	a GdkDevice to grab on.
block_others	`TRUE` to prevent other devices to interact with `widget` .

Since: 3.0

---

gtk_device_grab_remove ()

```
void
gtk_device_grab_remove (GtkWidget *widget,
                        GdkDevice *device);
```

Removes a device grab from the given widget.

You have to pair calls to `gtk_device_grab_add()` and `gtk_device_grab_remove()`.

Parameters

widget	a GtkWidget
device	a GdkDevice

Since: 3.0

---

gtk_key_snooper_install ()

```
guint
gtk_key_snooper_install (GtkKeySnoopFunc snooper,
                         gpointer func_data);
```

`gtk_key_snooper_install` has been deprecated since version 3.4 and should not be used in newly-written code.

Key snooping should not be done. Events should be handled by widgets.

Installs a key snooper function, which will get called on all key events before delivering them normally.

[skip]

Parameters

snooper	a GtkKeySnoopFunc	
func_data	data to pass to `snooper` .	[closure]

Returns

a unique id for this key snooper for use with `gtk_key_snooper_remove()`.

GtkKeySnoopFunc ()

```
gint
(*GtkKeySnoopFunc) (GtkWidget *grab_widget,
                    GdkEventKey *event,
                    gpointer func_data);
```

Key snooper functions are called before normal event delivery. They can be used to implement custom key event handling.

Parameters

grab_widget	the widget to which the event will be delivered	
event	the key event	
func_data	data supplied to `gtk_key_snooper_install()`.	[closure]

Returns

`TRUE` to stop further processing of `event` , `FALSE` to continue.

---

gtk_key_snooper_remove ()

```
void
gtk_key_snooper_remove (guint snooper_handler_id);
```

`gtk_key_snooper_remove` has been deprecated since version 3.4 and should not be used in newly-written code.

Key snooping should not be done. Events should be handled by widgets.

Removes the key snooper function with the given id.

Parameters

snooper_handler_id	Identifies the key snooper to remove

---

gtk_get_current_event ()

```
GdkEvent *
gtk_get_current_event (void);
```

Obtains a copy of the event currently being processed by GTK+.

For example, if you are handling a “clicked” signal, the current event will be the GdkEventButton that triggered the ::clicked signal.

Returns

a copy of the current event, or `NULL` if there is no current event. The returned event must be freed with `gdk_event_free()`.

[transfer full][nullable]

---

gtk_get_current_event_time ()

```
guint32
gtk_get_current_event_time (void);
```

If there is a current event and it has a timestamp, return that timestamp, otherwise return `GDK_CURRENT_TIME`.

Returns

the timestamp from the current event, or `GDK_CURRENT_TIME`.

---

gtk_get_current_event_state ()

```
gboolean
gtk_get_current_event_state (GdkModifierType *state);
```

If there is a current event and it has a state field, place that state field in `state` and return `TRUE`, otherwise return `FALSE`.

Parameters

state	a location to store the state of the current event.	[out]

Returns

`TRUE` if there was a current event and it had a state field

---

gtk_get_current_event_device ()

```
GdkDevice *
gtk_get_current_event_device (void);
```

If there is a current event and it has a device, return that device, otherwise return `NULL`.

Returns

a GdkDevice, or `NULL`.

[transfer none][nullable]

---

gtk_get_event_widget ()

```
GtkWidget *
gtk_get_event_widget (GdkEvent *event);
```

If `event` is `NULL` or the event was not associated with any widget, returns `NULL`, otherwise returns the widget that received the event originally.

Parameters

event	a GdkEvent

Returns

the widget that originally received `event` , or `NULL`.

[transfer none][nullable]

---

gtk_propagate_event ()

```
void
gtk_propagate_event (GtkWidget *widget,
                     GdkEvent *event);
```

Sends an event to a widget, propagating the event to parent widgets if the event remains unhandled.

Events received by GTK+ from GDK normally begin in `gtk_main_do_event()`. Depending on the type of event, existence of modal dialogs, grabs, etc., the event may be propagated; if so, this function is used.

gtk_propagate_event() calls `gtk_widget_event()` on each widget it decides to send the event to. So `gtk_widget_event()` is the lowest-level function; it simply emits the "event" and possibly an event-specific signal on a widget. `gtk_propagate_event()` is a bit higher-level, and `gtk_main_do_event()` is the highest level.

All that said, you most likely don't want to use any of these functions; synthesizing events is rarely needed. There are almost certainly better ways to achieve your goals. For example, use `gdk_window_invalidate_rect()` or `gtk_widget_queue_draw()` instead of making up expose events.

Parameters

widget	a GtkWidget
event	an event

Types and Values

GTK_PRIORITY_RESIZE

```
#define GTK_PRIORITY_RESIZE (G_PRIORITY_HIGH_IDLE + 10)
```

Use this priority for functionality related to size allocation.

It is used internally by GTK+ to compute the sizes of widgets. This priority is higher than `GDK_PRIORITY_REDRAW` to avoid resizing a widget which was just redrawn.

See Also

See the GLib manual, especially GMainLoop and signal-related functions such as `g_signal_connect()`

Version Information

Version Information — Variables and functions to check the GTK+ version

Functions

guint	gtk_get_major_version ()
guint	gtk_get_minor_version ()
guint	gtk_get_micro_version ()
guint	gtk_get_binary_age ()
guint	gtk_get_interface_age ()
const gchar *	gtk_check_version ()
#define	GTK_CHECK_VERSION()

Types and Values

#define	gtk_major_version
#define	gtk_minor_version
#define	gtk_micro_version
#define	gtk_binary_age
#define	gtk_interface_age
#define	GTK_MAJOR_VERSION
#define	GTK_MINOR_VERSION
#define	GTK_MICRO_VERSION
#define	GTK_BINARY_AGE
#define	GTK_INTERFACE_AGE

Includes

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

Description

GTK+ provides version information, primarily useful in configure checks for builds that have a configure script. Applications will not typically use the features described here.

Functions

gtk_get_major_version ()

```
guint
gtk_get_major_version (void);
```

Returns the major version number of the GTK+ library. (e.g. in GTK+ version 3.1.5 this is 3.)

This function is in the library, so it represents the GTK+ library your code is running against. Contrast with the GTK_MAJOR_VERSION macro, which represents the major version of the GTK+ headers you have included when compiling your code.

Returns

the major version number of the GTK+ library

Since: 3.0

gtk_get_minor_version ()

```
guint
gtk_get_minor_version (void);
```

Returns the minor version number of the GTK+ library. (e.g. in GTK+ version 3.1.5 this is 1.)

This function is in the library, so it represents the GTK+ library your code is are running against. Contrast with the GTK_MINOR_VERSION macro, which represents the minor version of the GTK+ headers you have included when compiling your code.

Returns

the minor version number of the GTK+ library

Since: 3.0

---

gtk_get_micro_version ()

```
guint
gtk_get_micro_version (void);
```

Returns the micro version number of the GTK+ library. (e.g. in GTK+ version 3.1.5 this is 5.)

This function is in the library, so it represents the GTK+ library your code is are running against. Contrast with the GTK_MICRO_VERSION macro, which represents the micro version of the GTK+ headers you have included when compiling your code.

Returns

the micro version number of the GTK+ library

Since: 3.0

---

gtk_get_binary_age ()

```
guint
gtk_get_binary_age (void);
```

Returns the binary age as passed to `libtool` when building the GTK+ library the process is running against. If `libtool` means nothing to you, don't worry about it.

Returns

the binary age of the GTK+ library

Since: 3.0

---

gtk_get_interface_age ()

```
guint
gtk_get_interface_age (void);
```

Returns the interface age as passed to `libtool` when building the GTK+ library the process is running against. If `libtool` means nothing to you, don't worry about it.

Returns

the interface age of the GTK+ library

Since: 3.0

---

gtk_check_version ()

```
const gchar *
gtk_check_version (guint required_major,
                   guint required_minor,
                   guint required_micro);
```

Checks that the GTK+ library in use is compatible with the given version. Generally you would pass in the constants GTK_MAJOR_VERSION, GTK_MINOR_VERSION, GTK_MICRO_VERSION as the three arguments to this function; that produces a check that the library in use is compatible with the version of GTK+ the application or module was compiled against.

Compatibility is defined by two things: first the version of the running library is newer than the version `required_major.required_minor.required_micro`. Second the running library must be binary compatible with the version `required_major.required_minor.required_micro` (same major version.)

This function is primarily for GTK+ modules; the module can call this function to check that it wasn't loaded into an incompatible version of GTK+. However, such a check isn't completely reliable, since the module may be linked against an old version of GTK+ and calling the old version of `gtk_check_version()`, but still get loaded into an application using a newer version of GTK+.

Parameters

required_major	the required major version
required_minor	the required minor version
required_micro	the required micro version

Returns

`NULL` if the GTK+ library is compatible with the given version, or a string describing the version mismatch. The returned string is owned by GTK+ and should not be modified or freed.

[nullable]

GTK_CHECK_VERSION()

```
#define             GTK_CHECK_VERSION(major,minor,micro)
```

Returns `TRUE` if the version of the GTK+ header files is the same as or newer than the passed-in version.

Parameters

major	major version (e.g. 1 for version 1.2.5)
minor	minor version (e.g. 2 for version 1.2.5)
micro	micro version (e.g. 5 for version 1.2.5)

Returns

`TRUE` if GTK+ headers are new enough

Types and Values

gtk_major_version

```
#define gtk_major_version gtk_get_major_version ()
```

---

gtk_minor_version

```
#define gtk_minor_version gtk_get_minor_version ()
```

---

gtk_micro_version

```
#define gtk_micro_version gtk_get_micro_version ()
```

---

gtk_binary_age

```
#define gtk_binary_age gtk_get_binary_age ()
```

---

gtk_interface_age

```
#define gtk_interface_age gtk_get_interface_age ()
```

GTK_MAJOR_VERSION

```
#define GTK_MAJOR_VERSION (3)
```

Like gtk_get_major_version(), but from the headers used at application compile time, rather than from the library linked against at application run time.

GTK_MINOR_VERSION

```
#define GTK_MINOR_VERSION (24)
```

Like gtk_get_minor_version(), but from the headers used at application compile time, rather than from the library linked against at application run time.

GTK_MICRO_VERSION

```
#define GTK_MICRO_VERSION (10)
```

Like gtk_get_micro_version(), but from the headers used at application compile time, rather than from the library linked against at application run time.

GTK_BINARY_AGE

```
#define GTK_BINARY_AGE    (2410)
```

Like gtk_get_binary_age(), but from the headers used at application compile time, rather than from the library linked against at application run time.

GTK_INTERFACE_AGE

```
#define GTK_INTERFACE_AGE (6)
```

Like gtk_get_interface_age(), but from the headers used at application compile time, rather than from the library linked against at application run time.

Accelerator Groups

Accelerator Groups — Groups of global keyboard accelerators for an entire GtkWindow

Functions

GtkAccelGroup *	gtk_accel_group_new ()
void	gtk_accel_group_connect ()
void	gtk_accel_group_connect_by_path ()
gboolean	(*GtkAccelGroupActivate) ()
gboolean	(*GtkAccelGroupFindFunc) ()
gboolean	gtk_accel_group_disconnect ()
gboolean	gtk_accel_group_disconnect_key ()
gboolean	gtk_accel_group_activate ()
void	gtk_accel_group_lock ()
void	gtk_accel_group_unlock ()
gboolean	gtk_accel_group_get_is_locked ()
GtkAccelGroup *	gtk_accel_group_from_accel_closure ()
GdkModifierType	gtk_accel_group_get_modifier_mask ()
gboolean	gtk_accel_groups_activate ()
GSList *	gtk_accel_groups_from_object ()
GtkAccelKey *	gtk_accel_group_find ()
gboolean	gtk_accelerator_valid ()
void	gtk_accelerator_parse ()
gchar *	gtk_accelerator_name ()
gchar *	gtk_accelerator_get_label ()
void	gtk_accelerator_parse_with_keycode ()
gchar *	gtk_accelerator_name_with_keycode ()
gchar *	gtk_accelerator_get_label_with_keycode ()
void	gtk_accelerator_set_default_mod_mask ()
GdkModifierType	gtk_accelerator_get_default_mod_mask ()

Properties

gboolean	is-locked	Read
GdkModifierType	modifier-mask	Read

Signals

gboolean	accel-activate	Has Details
void	accel-changed	Has Details

Types and Values

struct	GtkAccelGroup
struct	GtkAccelGroupClass
enum	GtkAccelFlags
struct	GtkAccelKey

Object Hierarchy

```
GObject
└── GtkAccelGroup
```

Includes

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

Description

A GtkAccelGroup represents a group of keyboard accelerators, typically attached to a toplevel GtkWindow (with `gtk_window_add_accel_group()`). Usually you won't need to create a GtkAccelGroup directly; instead, when using GtkUIManager, GTK+ automatically sets up the accelerators for your menus in the ui manager's GtkAccelGroup.

Note that "accelerators" are different from "mnemonics". Accelerators are shortcuts for activating a menu item; they appear alongside the menu item they're a shortcut for. For example "Ctrl+Q" might appear alongside the "Quit" menu item. Mnemonics are shortcuts for GUI elements such as text entries or buttons; they appear as underlined characters. See `gtk_label_new_with_mnemonic()`. Menu items can have both accelerators and mnemonics, of course.

Functions

gtk_accel_group_new ()

```
GtkAccelGroup *
gtk_accel_group_new (void);
```

Creates a new GtkAccelGroup.

Returns

a new GtkAccelGroup object

---

gtk_accel_group_connect ()

```
void
gtk_accel_group_connect (GtkAccelGroup *accel_group,
                         guint accel_key,
                         GdkModifierType accel_mods,
                         GtkAccelFlags accel_flags,
                         GClosure *closure);
```

Installs an accelerator in this group. When `accel_group` is being activated in response to a call to `gtk_accel_groups_activate()`, `closure` will be invoked if the `accel_key` and `accel_mods` from `gtk_accel_groups_activate()` match those of this connection.

The signature used for the `closure` is that of GtkAccelGroupActivate.

Note that, due to implementation details, a single closure can only be connected to one accelerator group.

Parameters

accel_group	the accelerator group to install an accelerator in
accel_key	key value of the accelerator
accel_mods	modifier combination of the accelerator
accel_flags	a flag mask to configure this accelerator
closure	closure to be executed upon accelerator activation

---

gtk_accel_group_connect_by_path ()

```
void
gtk_accel_group_connect_by_path (GtkAccelGroup *accel_group,
                                 const gchar *accel_path,
                                 GClosure *closure);
```

Installs an accelerator in this group, using an accelerator path to look up the appropriate key and modifiers (see `gtk_accel_map_add_entry()`). When `accel_group` is being activated in response to a call to `gtk_accel_groups_activate()`, `closure` will be invoked if the `accel_key` and `accel_mods` from `gtk_accel_groups_activate()` match the key and modifiers for the path.

The signature used for the `closure` is that of GtkAccelGroupActivate.

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

accel_group	the accelerator group to install an accelerator in
accel_path	path used for determining key and modifiers
closure	closure to be executed upon accelerator activation

GtkAccelGroupActivate ()

```
gboolean
(*GtkAccelGroupActivate) (GtkAccelGroup *accel_group,
                          GObject *acceleratable,
                          guint keyval,
                          GdkModifierType modifier);
```

GtkAccelGroupFindFunc ()

```
gboolean
(*GtkAccelGroupFindFunc) (GtkAccelKey *key,
                          GClosure *closure,
                          gpointer data);
```

Parameters

data	.	[closure]

Since: 2.2

---

gtk_accel_group_disconnect ()

```
gboolean
gtk_accel_group_disconnect (GtkAccelGroup *accel_group,
                            GClosure *closure);
```

Removes an accelerator previously installed through `gtk_accel_group_connect()`.

Since 2.20 `closure` can be `NULL`.

Parameters

accel_group	the accelerator group to remove an accelerator from	
closure	the closure to remove from this accelerator group, or `NULL` to remove all closures.	[allow-none]

Returns

`TRUE` if the closure was found and got disconnected

---

gtk_accel_group_disconnect_key ()

```
gboolean
gtk_accel_group_disconnect_key (GtkAccelGroup *accel_group,
                                guint accel_key,
                                GdkModifierType accel_mods);
```

Removes an accelerator previously installed through `gtk_accel_group_connect()`.

Parameters

accel_group	the accelerator group to install an accelerator in
accel_key	key value of the accelerator
accel_mods	modifier combination of the accelerator

Returns

`TRUE` if there was an accelerator which could be removed, `FALSE` otherwise

---

gtk_accel_group_activate ()

```
gboolean
gtk_accel_group_activate (GtkAccelGroup *accel_group,
                          GQuark accel_quark,
                          GObject *acceleratable,
                          guint accel_key,
                          GdkModifierType accel_mods);
```

Finds the first accelerator in `accel_group` that matches `accel_key` and `accel_mods` , and activates it.

Parameters

accel_group	a GtkAccelGroup
accel_quark	the quark for the accelerator name
acceleratable	the GObject, usually a GtkWindow, on which to activate the accelerator
accel_key	accelerator keyval from a key event
accel_mods	keyboard state mask from a key event

Returns

`TRUE` if an accelerator was activated and handled this keypress

---

gtk_accel_group_lock ()

```
void
gtk_accel_group_lock (GtkAccelGroup *accel_group);
```

Locks the given accelerator group.

Locking an acelerator group prevents the accelerators contained within it to be changed during runtime. Refer to `gtk_accel_map_change_entry()` about runtime accelerator changes.

If called more than once, `accel_group` remains locked until `gtk_accel_group_unlock()` has been called an equivalent number of times.

Parameters

accel_group	a GtkAccelGroup

---

gtk_accel_group_unlock ()

```
void
gtk_accel_group_unlock (GtkAccelGroup *accel_group);
```

Undoes the last call to `gtk_accel_group_lock()` on this `accel_group` .

Parameters

accel_group	a GtkAccelGroup

---

gtk_accel_group_get_is_locked ()

```
gboolean
gtk_accel_group_get_is_locked (GtkAccelGroup *accel_group);
```

Locks are added and removed using `gtk_accel_group_lock()` and `gtk_accel_group_unlock()`.

Parameters

accel_group	a GtkAccelGroup

Returns

`TRUE` if there are 1 or more locks on the `accel_group` , `FALSE` otherwise.

Since: 2.14

---

gtk_accel_group_from_accel_closure ()

```
GtkAccelGroup *
gtk_accel_group_from_accel_closure (GClosure *closure);
```

Finds the GtkAccelGroup to which `closure` is connected; see `gtk_accel_group_connect()`.

Parameters

closure	a GClosure

Returns

the GtkAccelGroup to which `closure` is connected, or `NULL`.

[nullable][transfer none]

---

gtk_accel_group_get_modifier_mask ()

```
GdkModifierType
gtk_accel_group_get_modifier_mask (GtkAccelGroup *accel_group);
```

Gets a GdkModifierType representing the mask for this `accel_group` . For example, GDK_CONTROL_MASK, GDK_SHIFT_MASK, etc.

Parameters

accel_group	a GtkAccelGroup

Returns

the modifier mask for this accel group.

Since: 2.14

---

gtk_accel_groups_activate ()

```
gboolean
gtk_accel_groups_activate (GObject *object,
                           guint accel_key,
                           GdkModifierType accel_mods);
```

Finds the first accelerator in any GtkAccelGroup attached to `object` that matches `accel_key` and `accel_mods` , and activates that accelerator.

Parameters

object	the GObject, usually a GtkWindow, on which to activate the accelerator
accel_key	accelerator keyval from a key event
accel_mods	keyboard state mask from a key event

Returns

`TRUE` if an accelerator was activated and handled this keypress

---

gtk_accel_groups_from_object ()

```
GSList *
gtk_accel_groups_from_object (GObject *object);
```

Gets a list of all accel groups which are attached to `object` .

Parameters

object	a GObject, usually a GtkWindow

Returns

a list of all accel groups which are attached to `object` .

[element-type GtkAccelGroup][transfer none]

---

gtk_accel_group_find ()

```
GtkAccelKey *
gtk_accel_group_find (GtkAccelGroup *accel_group,
                      GtkAccelGroupFindFunc find_func,
                      gpointer data);
```

Finds the first entry in an accelerator group for which `find_func` returns `TRUE` and returns its GtkAccelKey.

Parameters

accel_group	a GtkAccelGroup	
find_func	a function to filter the entries of `accel_group` with.	[scope call]
data	data to pass to `find_func`	

Returns

the key of the first entry passing `find_func` . The key is owned by GTK+ and must not be freed.

[transfer none]

---

gtk_accelerator_valid ()

```
gboolean
gtk_accelerator_valid (guint keyval,
                       GdkModifierType modifiers);
```

Determines whether a given keyval and modifier mask constitute a valid keyboard accelerator. For example, the GDK_KEY_a keyval plus GDK_CONTROL_MASK is valid - this is a “Ctrl+a” accelerator. But, you can't, for instance, use the GDK_KEY_Control_L keyval as an accelerator.

Parameters

keyval	a GDK keyval
modifiers	modifier mask

Returns

`TRUE` if the accelerator is valid

---

gtk_accelerator_parse ()

```
void
gtk_accelerator_parse (const gchar *accelerator,
                       guint *accelerator_key,
                       GdkModifierType *accelerator_mods);
```

Parses a string representing an accelerator. The format looks like “<Control>a” or “<Shift><Alt>F1” or “<Release>z” (the last one is for key release).

The parser is fairly liberal and allows lower or upper case, and also abbreviations such as “<Ctl>” and “<Ctrl>”. Key names are parsed using `gdk_keyval_from_name()`. For character keys the name is not the symbol, but the lowercase name, e.g. one would use “<Ctrl>minus” instead of “<Ctrl>-”.

If the parse fails, `accelerator_key` and `accelerator_mods` will be set to 0 (zero).

Parameters

accelerator	string representing an accelerator	
accelerator_key	return location for accelerator keyval, or `NULL`.	[out][allow-none]
accelerator_mods	return location for accelerator modifier mask, `NULL`.	[out][allow-none]

---

gtk_accelerator_name ()

```
gchar *
gtk_accelerator_name (guint accelerator_key,
                      GdkModifierType accelerator_mods);
```

Converts an accelerator keyval and modifier mask into a string parseable by `gtk_accelerator_parse()`. For example, if you pass in GDK_KEY_q and GDK_CONTROL_MASK, this function returns “<Control>q”.

If you need to display accelerators in the user interface, see `gtk_accelerator_get_label()`.

Parameters

accelerator_key	accelerator keyval
accelerator_mods	accelerator modifier mask

Returns

a newly-allocated accelerator name

---

gtk_accelerator_get_label ()

```
gchar *
gtk_accelerator_get_label (guint accelerator_key,
                           GdkModifierType accelerator_mods);
```

Converts an accelerator keyval and modifier mask into a string which can be used to represent the accelerator to the user.

Parameters

accelerator_key	accelerator keyval
accelerator_mods	accelerator modifier mask

Returns

a newly-allocated string representing the accelerator.

Since: 2.6

---

gtk_accelerator_parse_with_keycode ()

```
void
gtk_accelerator_parse_with_keycode (const gchar *accelerator,
                                    guint *accelerator_key,
                                    guint **accelerator_codes,
                                    GdkModifierType *accelerator_mods);
```

Parses a string representing an accelerator, similarly to `gtk_accelerator_parse()` but handles keycodes as well. This is only useful for system-level components, applications should use `gtk_accelerator_parse()` instead.

If `accelerator_codes` is given and the result stored in it is non-`NULL`, the result must be freed with `g_free()`.

If a keycode is present in the accelerator and no `accelerator_codes` is given, the parse will fail.

If the parse fails, `accelerator_key` , `accelerator_mods` and `accelerator_codes` will be set to 0 (zero).

Parameters

accelerator	string representing an accelerator	
accelerator_key	return location for accelerator keyval, or `NULL`.	[out][allow-none]
accelerator_codes	return location for accelerator keycodes, or `NULL`.	[out][array zero-terminated=1][transfer full][allow-none]
accelerator_mods	return location for accelerator modifier mask, `NULL`.	[out][allow-none]

Since: 3.4

---

gtk_accelerator_name_with_keycode ()

```
gchar *
gtk_accelerator_name_with_keycode (GdkDisplay *display,
                                   guint accelerator_key,
                                   guint keycode,
                                   GdkModifierType accelerator_mods);
```

Converts an accelerator keyval and modifier mask into a string parseable by `gtk_accelerator_parse_with_keycode()`, similarly to `gtk_accelerator_name()` but handling keycodes. This is only useful for system-level components, applications should use `gtk_accelerator_parse()` instead.

Parameters

display	a GdkDisplay or `NULL` to use the default display.	[allow-none]
accelerator_key	accelerator keyval	
keycode	accelerator keycode	
accelerator_mods	accelerator modifier mask	

Returns

a newly allocated accelerator name.

Since: 3.4

---

gtk_accelerator_get_label_with_keycode ()

```
gchar *
gtk_accelerator_get_label_with_keycode
                                (GdkDisplay *display,
                                 guint accelerator_key,
                                 guint keycode,
                                 GdkModifierType accelerator_mods);
```

Converts an accelerator keyval and modifier mask into a (possibly translated) string that can be displayed to a user, similarly to `gtk_accelerator_get_label()`, but handling keycodes.

This is only useful for system-level components, applications should use `gtk_accelerator_parse()` instead.

Parameters

display	a GdkDisplay or `NULL` to use the default display.	[allow-none]
accelerator_key	accelerator keyval	
keycode	accelerator keycode	
accelerator_mods	accelerator modifier mask	

Returns

a newly-allocated string representing the accelerator.

Since: 3.4

---

gtk_accelerator_set_default_mod_mask ()

```
void
gtk_accelerator_set_default_mod_mask (GdkModifierType default_mod_mask);
```

Sets the modifiers that will be considered significant for keyboard accelerators. The default mod mask depends on the GDK backend in use, but will typically include GDK_CONTROL_MASK | GDK_SHIFT_MASK | GDK_MOD1_MASK | GDK_SUPER_MASK | GDK_HYPER_MASK | GDK_META_MASK. In other words, Control, Shift, Alt, Super, Hyper and Meta. Other modifiers will by default be ignored by GtkAccelGroup.

You must include at least the three modifiers Control, Shift and Alt in any value you pass to this function.

The default mod mask should be changed on application startup, before using any accelerator groups.

Parameters

default_mod_mask	accelerator modifier mask

---

gtk_accelerator_get_default_mod_mask ()

```
GdkModifierType
gtk_accelerator_get_default_mod_mask (void);
```

Gets the modifier mask.

The modifier mask determines which modifiers are considered significant for keyboard accelerators. See `gtk_accelerator_set_default_mod_mask()`.

Returns

the default accelerator modifier mask

Types and Values

struct GtkAccelGroup

```
struct GtkAccelGroup;
```

An object representing and maintaining a group of accelerators.

---

struct GtkAccelGroupClass

```
struct GtkAccelGroupClass {
  GObjectClass parent_class;


  void (*accel_changed) (GtkAccelGroup *accel_group,
                         guint           keyval,
                         GdkModifierType modifier,
                         GClosure       *accel_closure);
};
```

Members

`accel_changed` ()	Signal emitted when an entry is added to or removed from the accel group.

---

enum GtkAccelFlags

Accelerator flags used with `gtk_accel_group_connect()`.

Members

GTK_ACCEL_VISIBLE	Accelerator is visible
GTK_ACCEL_LOCKED	Accelerator not removable
GTK_ACCEL_MASK	Mask

---

struct GtkAccelKey

```
struct GtkAccelKey {
  guint           accel_key;
  GdkModifierType accel_mods;
  guint           accel_flags : 16;
};
```

Members

guint `accel_key`;	The accelerator keyval
GdkModifierType `accel_mods`;	The accelerator modifiers
guint `accel_flags` : 16;	The accelerator flags

Property Details

The “is-locked” property

“is-locked” gboolean

Is the accel group locked.

Flags: Read

Default value: FALSE

The “modifier-mask” property

“modifier-mask” GdkModifierType

Modifier Mask.

Flags: Read

Signal Details

The “accel-activate” signal

```
gboolean
user_function (GtkAccelGroup  *accel_group,
               GObject        *acceleratable,
               guint           keyval,
               GdkModifierType modifier,
               gpointer        user_data)
```

The accel-activate signal is an implementation detail of GtkAccelGroup and not meant to be used by applications.

Parameters

accel_group	the GtkAccelGroup which received the signal
acceleratable	the object on which the accelerator was activated
keyval	the accelerator keyval
modifier	the modifier combination of the accelerator
user_data	user data set when the signal handler was connected.

Returns

TRUE if the accelerator was activated

Flags: Has Details

The "accel-changed" signal

```
void
user_function (GtkAccelGroup  *accel_group,
               guint           keyval,
               GdkModifierType modifier,
               GClosure       *accel_closure,
               gpointer        user_data)
```

The accel-changed signal is emitted when an entry is added to or removed from the accel group.

Widgets like GtkAccelLabel which display an associated accelerator should connect to this signal, and rebuild their visual representation if the `accel_closure` is theirs.

Parameters

accel_group	the GtkAccelGroup which received the signal
keyval	the accelerator keyval
modifier	the modifier combination of the accelerator
accel_closure	the GClosure of the accelerator
user_data	user data set when the signal handler was connected.

Flags: Has Details

See Also

gtk_window_add_accel_group(), `gtk_accel_map_change_entry()`, `gtk_label_new_with_mnemonic()`

Accelerator Maps

Accelerator Maps — Loadable keyboard accelerator specifications

Functions

void	(*GtkAccelMapForeach) ()
void	gtk_accel_map_add_entry ()
gboolean	gtk_accel_map_lookup_entry ()
gboolean	gtk_accel_map_change_entry ()
void	gtk_accel_map_load ()
void	gtk_accel_map_save ()
void	gtk_accel_map_foreach ()
void	gtk_accel_map_load_fd ()
void	gtk_accel_map_save_fd ()
void	gtk_accel_map_load_scanner ()
void	gtk_accel_map_add_filter ()
void	gtk_accel_map_foreach_unfiltered ()
GtkAccelMap *	gtk_accel_map_get ()
void	gtk_accel_map_lock_path ()

void	gtk_accel_map_unlock_path ()

Signals

void	changed	Has Details

Types and Values

GtkAccelMap

Object Hierarchy

```
GObject
└── GtkAccelMap
```

Includes

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

Description

Accelerator maps are used to define runtime configurable accelerators. Functions for manipulating them are are usually used by higher level convenience mechanisms like GtkUIManager and are thus considered “low-level”. You’ll want to use them if you’re manually creating menus that should have user-configurable accelerators.

An accelerator is uniquely defined by:

- accelerator path
- accelerator key
- accelerator modifiers

The accelerator path must consist of “<WINDOWTYPE>/Category1/Category2/.../Action”, where WINDOWTYPE should be a unique application-specific identifier that corresponds to the kind of window the accelerator is being used in, e.g. “Gimp-Image”, “Abiword-Document” or “Gnumeric-Settings”. The “Category1/.../Action” portion is most appropriately chosen by the action the accelerator triggers, i.e. for accelerators on menu items, choose the item’s menu path, e.g. “File/Save As”, “Image/View/Zoom” or “Edit/Select All”. So a full valid accelerator path may look like: “<Gimp-Toolbox>/File/Dialogs/Tool Options...”.

All accelerators are stored inside one global GtkAccelMap that can be obtained using `gtk_accel_map_get()`. See Monitoring changes for additional details.

Manipulating accelerators

New accelerators can be added using `gtk_accel_map_add_entry()`. To search for specific accelerator, use `gtk_accel_map_lookup_entry()`. Modifications of existing accelerators should be done using

gtk_accel_map_change_entry().

In order to avoid having some accelerators changed, they can be locked using gtk_accel_map_lock_path(). Unlocking is done using gtk_accel_map_unlock_path().

Saving and loading accelerator maps

Accelerator maps can be saved to and loaded from some external resource. For simple saving and loading from file, gtk_accel_map_save() and gtk_accel_map_load() are provided. Saving and loading can also be done by providing file descriptor to gtk_accel_map_save_fd() and gtk_accel_map_load_fd().

Monitoring changes

GtkAccelMap object is only useful for monitoring changes of accelerators. By connecting to "changed" signal, one can monitor changes of all accelerators. It is also possible to monitor only single accelerator path by using it as a detail of the "changed" signal.

Functions

GtkAccelMapForeach ()

```
void
(*GtkAccelMapForeach) (gpointer data,
                       const gchar *accel_path,
                       guint accel_key,
                       GdkModifierType accel_mods,
                       gboolean changed);
```

Parameters

data	User data passed to gtk_accel_map_foreach() or gtk_accel_map_foreach_unfiltered()
accel_path	Accel path of the current accelerator
accel_key	Key of the current accelerator
accel_mods	Modifiers of the current accelerator
changed	Changed flag of the accelerator (if TRUE, accelerator has changed during runtime and would need to be saved during an accelerator dump)

---

gtk_accel_map_add_entry ()

```
void
gtk_accel_map_add_entry (const gchar *accel_path,
                         guint accel_key,
```

```
                                GdkModifierType accel_mods);
```

Registers a new accelerator with the global accelerator map. This function should only be called once per `accel_path` with the canonical `accel_key` and `accel_mods` for this path. To change the accelerator during runtime programatically, use `gtk_accel_map_change_entry()`.

Set `accel_key` and `accel_mods` to 0 to request a removal of the accelerator.

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

accel_path	valid accelerator path
accel_key	the accelerator key
accel_mods	the accelerator modifiers

---

gtk_accel_map_lookup_entry ()

```
gboolean
gtk_accel_map_lookup_entry (const gchar *accel_path,
                            GtkAccelKey *key);
```

Looks up the accelerator entry for `accel_path` and fills in `key` .

Parameters

accel_path	a valid accelerator path	
key	the accelerator key to be filled in (optional).	[allow-none][out]

Returns

`TRUE` if `accel_path` is known, `FALSE` otherwise

---

gtk_accel_map_change_entry ()

```
gboolean
gtk_accel_map_change_entry (const gchar *accel_path,
                            guint accel_key,
                            GdkModifierType accel_mods,
                            gboolean replace);
```

Changes the `accel_key` and `accel_mods` currently associated with `accel_path` . Due to conflicts with other accelerators, a change may not always be possible, `replace` indicates whether other accelerators may be deleted to resolve such conflicts. A change will only occur if all conflicts could be resolved (which might not be the case if conflicting accelerators are locked). Successful changes are indicated by a `TRUE` return value.

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

accel_path	a valid accelerator path
accel_key	the new accelerator key
accel_mods	the new accelerator modifiers
replace	`TRUE` if other accelerators may be deleted upon conflicts

Returns

`TRUE` if the accelerator could be changed, `FALSE` otherwise

---

gtk_accel_map_load ()

```
void
gtk_accel_map_load (const gchar *file_name);
```

Parses a file previously saved with `gtk_accel_map_save()` for accelerator specifications, and propagates them accordingly.

Parameters

file_name	a file containing accelerator specifications, in the GLib file name encoding.	[type filename]

---

gtk_accel_map_save ()

```
void
gtk_accel_map_save (const gchar *file_name);
```

Saves current accelerator specifications (accelerator path, key and modifiers) to `file_name` . The file is written in a format suitable to be read back in by `gtk_accel_map_load()`.

Parameters

file_name	the name of the file to contain accelerator specifications, in the GLib file name encoding.	[type filename]

---

gtk_accel_map_foreach ()

```
void
gtk_accel_map_foreach (gpointer data,
                       GtkAccelMapForeach foreach_func);
```

Loops over the entries in the accelerator map whose accel path doesn’t match any of the filters added with `gtk_accel_map_add_filter()`, and execute `foreach_func` on each. The signature of `foreach_func` is that of

GtkAccelMapForeach, the `changed` parameter indicates whether this accelerator was changed during runtime (thus, would need saving during an accelerator map dump).

Parameters

data	data to be passed into `foreach_func` .	[allow-none]
foreach_func	function to be executed for each accel map entry which is not filtered out.	[scope call]

gtk_accel_map_load_fd ()

```
void
gtk_accel_map_load_fd (gint fd);
```

Filedescriptor variant of `gtk_accel_map_load()`.

Note that the file descriptor will not be closed by this function.

Parameters

fd	a valid readable file descriptor

gtk_accel_map_save_fd ()

```
void
gtk_accel_map_save_fd (gint fd);
```

Filedescriptor variant of `gtk_accel_map_save()`.

Note that the file descriptor will not be closed by this function.

Parameters

fd	a valid writable file descriptor

gtk_accel_map_load_scanner ()

```
void
gtk_accel_map_load_scanner (GScanner *scanner);
```

GScanner variant of `gtk_accel_map_load()`.

Parameters

scanner	a GScanner which has already been provided with an input file

gtk_accel_map_add_filter ()

```
void
gtk_accel_map_add_filter (const gchar *filter_pattern);
```

Adds a filter to the global list of accel path filters.

Accel map entries whose accel path matches one of the filters are skipped by `gtk_accel_map_foreach()`.

This function is intended for GTK+ modules that create their own menus, but don't want them to be saved into the applications accelerator map dump.

Parameters

filter_pattern	a pattern (see GPatternSpec)

---

gtk_accel_map_foreach_unfiltered ()

```
void
gtk_accel_map_foreach_unfiltered (gpointer data,
                                  GtkAccelMapForeach foreach_func);
```

Loops over all entries in the accelerator map, and execute `foreach_func` on each. The signature of `foreach_func` is that of GtkAccelMapForeach, the `changed` parameter indicates whether this accelerator was changed during runtime (thus, would need saving during an accelerator map dump).

Parameters

data	data to be passed into `foreach_func`	
foreach_func	function to be executed for each accel map entry.	[scope call]

---

gtk_accel_map_get ()

```
GtkAccelMap *
gtk_accel_map_get (void);
```

Gets the singleton global GtkAccelMap object. This object is useful only for notification of changes to the accelerator map via the ::changed signal; it isn't a parameter to the other accelerator map functions.

Returns

the global GtkAccelMap object.

[transfer none]

Since: 2.4

---

gtk_accel_map_lock_path ()

```
void
gtk_accel_map_lock_path (const gchar *accel_path);
```

Locks the given accelerator path. If the accelerator map doesn't yet contain an entry for `accel_path` , a new one is created.

Locking an accelerator path prevents its accelerator from being changed during runtime. A locked accelerator path can be unlocked by `gtk_accel_map_unlock_path()`. Refer to `gtk_accel_map_change_entry()` for information about runtime accelerator changes.

If called more than once, `accel_path` remains locked until `gtk_accel_map_unlock_path()` has been called an equivalent number of times.

Note that locking of individual accelerator paths is independent from locking the GtkAccelGroup containing them. For runtime accelerator changes to be possible, both the accelerator path and its GtkAccelGroup have to be unlocked.

Parameters

accel_path	a valid accelerator path

Since: 2.4

---

gtk_accel_map_unlock_path ()

```
void
gtk_accel_map_unlock_path (const gchar *accel_path);
```

Undoes the last call to `gtk_accel_map_lock_path()` on this `accel_path` . Refer to `gtk_accel_map_lock_path()` for information about accelerator path locking.

Parameters

accel_path	a valid accelerator path

Since: 2.4

Types and Values

GtkAccelMap

```
typedef struct _GtkAccelMap GtkAccelMap;
```

Signal Details

The "changed" signal

```
void
user_function (GtkAccelMap    *object,
               gchar          *accel_path,
               guint           accel_key,
               GdkModifierType accel_mods,
               gpointer        user_data)
```

Notifies of a change in the global accelerator map. The path is also used as the detail for the signal, so it is possible to connect to changed::`accel_path`.

Parameters

object	the global accel map object
accel_path	the path of the accelerator that changed
accel_key	the key value for the new accelerator
accel_mods	the modifier mask for the new accelerator
user_data	user data set when the signal handler was connected.

Flags: Has Details

Since: 2.4

See Also

GtkAccelGroup, GtkAccelKey, GtkUIManager, `gtk_widget_set_accel_path()`, `gtk_menu_item_set_accel_path()`

Clipboards

Clipboards — Storing data on clipboards

Functions

void	(*GtkClipboardReceivedFunc) ()
void	(*GtkClipboardTextReceivedFunc) ()
void	(*GtkClipboardImageReceivedFunc) ()
void	(*GtkClipboardTargetsReceivedFunc) ()
void	(*GtkClipboardRichTextReceivedFunc) ()
void	(*GtkClipboardURIReceivedFunc) ()
void	(*GtkClipboardGetFunc) ()
void	(*GtkClipboardClearFunc) ()
GtkClipboard *	gtk_clipboard_get ()
GtkClipboard *	gtk_clipboard_get_for_display ()
GdkDisplay *	gtk_clipboard_get_display ()
GtkClipboard *	gtk_clipboard_get_default ()
gboolean	gtk_clipboard_set_with_data ()
gboolean	gtk_clipboard_set_with_owner ()
GObject *	gtk_clipboard_get_owner ()

void	gtk_clipboard_clear ()
void	gtk_clipboard_set_text ()
void	gtk_clipboard_set_image ()
void	gtk_clipboard_request_contents ()
void	gtk_clipboard_request_text ()
void	gtk_clipboard_request_image ()
void	gtk_clipboard_request_targets ()
void	gtk_clipboard_request_rich_text ()
void	gtk_clipboard_request_uris ()
GtkSelectionData *	gtk_clipboard_wait_for_contents ()
gchar *	gtk_clipboard_wait_for_text ()
GdkPixbuf *	gtk_clipboard_wait_for_image ()
guint8 *	gtk_clipboard_wait_for_rich_text ()
gchar **	gtk_clipboard_wait_for_uris ()
gboolean	gtk_clipboard_wait_is_text_available ()
gboolean	gtk_clipboard_wait_is_image_available ()
gboolean	gtk_clipboard_wait_is_rich_text_available ()
gboolean	gtk_clipboard_wait_is_uris_available ()
gboolean	gtk_clipboard_wait_for_targets ()
gboolean	gtk_clipboard_wait_is_target_available ()
void	gtk_clipboard_set_can_store ()
void	gtk_clipboard_store ()
GdkAtom	gtk_clipboard_get_selection ()

Signals

void	owner-change	Run First

Types and Values

GtkClipboard

Object Hierarchy

```
GObject
└── GtkClipboard
```

Includes

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

Description

The GtkClipboard object represents a clipboard of data shared between different processes or between different widgets in the same process. Each clipboard is identified by a name encoded as a GdkAtom. (Conversion to and from strings can be done with `gdk_atom_intern()` and `gdk_atom_name()`.) The default clipboard corresponds

to the “CLIPBOARD” atom; another commonly used clipboard is the “PRIMARY” clipboard, which, in X, traditionally contains the currently selected text.

To support having a number of different formats on the clipboard at the same time, the clipboard mechanism allows providing callbacks instead of the actual data. When you set the contents of the clipboard, you can either supply the data directly (via functions like `gtk_clipboard_set_text()`), or you can supply a callback to be called at a later time when the data is needed (via `gtk_clipboard_set_with_data()` or `gtk_clipboard_set_with_owner()`.) Providing a callback also avoids having to make copies of the data when it is not needed.

gtk_clipboard_set_with_data() and `gtk_clipboard_set_with_owner()` are quite similar; the choice between the two depends mostly on which is more convenient in a particular situation. The former is most useful when you want to have a blob of data with callbacks to convert it into the various data types that you advertise. When the `clear_func` you provided is called, you simply free the data blob. The latter is more useful when the contents of clipboard reflect the internal state of a GObject (As an example, for the PRIMARY clipboard, when an entry widget provides the clipboard’s contents the contents are simply the text within the selected region.) If the contents change, the entry widget can call `gtk_clipboard_set_with_owner()` to update the timestamp for clipboard ownership, without having to worry about `clear_func` being called.

Requesting the data from the clipboard is essentially asynchronous. If the contents of the clipboard are provided within the same process, then a direct function call will be made to retrieve the data, but if they are provided by another process, then the data needs to be retrieved from the other process, which may take some time. To avoid blocking the user interface, the call to request the selection, `gtk_clipboard_request_contents()` takes a callback that will be called when the contents are received (or when the request fails.) If you don’t want to deal with providing a separate callback, you can also use `gtk_clipboard_wait_for_contents()`. What this does is run the GLib main loop recursively waiting for the contents. This can simplify the code flow, but you still have to be aware that other callbacks in your program can be called while this recursive mainloop is running.

Along with the functions to get the clipboard contents as an arbitrary data chunk, there are also functions to retrieve it as text, `gtk_clipboard_request_text()` and `gtk_clipboard_wait_for_text()`. These functions take care of determining which formats are advertised by the clipboard provider, asking for the clipboard in the best available format and converting the results into the UTF-8 encoding. (The standard form for representing strings in GTK+.)

Functions

GtkClipboardReceivedFunc ()

```
void
(*GtkClipboardReceivedFunc) (GtkClipboard *clipboard,
                             GtkSelectionData *selection_data,
                             gpointer data);
```

A function to be called when the results of `gtk_clipboard_request_contents()` are received, or when the request fails.

Parameters

clipboard	the GtkClipboard

selection_data	a GtkSelectionData containing the data was received. If retrieving the data failed, then then length field of `selection_data` will be negative.	
data	the `user_data` supplied to `gtk_clipboard_request_contents()`.	[closure]

GtkClipboardTextReceivedFunc ()

```
void
(*GtkClipboardTextReceivedFunc) (GtkClipboard *clipboard,
                                 const gchar *text,
                                 gpointer data);
```

A function to be called when the results of `gtk_clipboard_request_text()` are received, or when the request fails.

Parameters

clipboard	the GtkClipboard	
text	the text received, as a UTF-8 encoded string, or `NULL` if retrieving the data failed.	[nullable]
data	the `user_data` supplied to `gtk_clipboard_request_text()`.	[closure]

GtkClipboardImageReceivedFunc ()

```
void
(*GtkClipboardImageReceivedFunc) (GtkClipboard *clipboard,
                                  GdkPixbuf *pixbuf,
                                  gpointer data);
```

A function to be called when the results of `gtk_clipboard_request_image()` are received, or when the request fails.

Parameters

clipboard	the GtkClipboard	
pixbuf	the received image	
data	the `user_data` supplied to `gtk_clipboard_request_image()`.	[closure]

Since: 2.6

GtkClipboardTargetsReceivedFunc ()

```
void
(*GtkClipboardTargetsReceivedFunc) (GtkClipboard *clipboard,
                                    GdkAtom *atoms,
                                    gint n_atoms,
                                    gpointer data);
```

A function to be called when the results of `gtk_clipboard_request_targets()` are received, or when the request fails.

Parameters

clipboard	the GtkClipboard	
atoms	the supported targets, as array of GdkAtom, or `NULL` if retrieving the data failed.	[nullable][array length=n_atoms]
n_atoms	the length of the `atoms` array.	
data	the `user_data` supplied to `gtk_clipboard_request_targets()`.	[closure]

Since: 2.4

GtkClipboardRichTextReceivedFunc ()

```
void
(*GtkClipboardRichTextReceivedFunc) (GtkClipboard *clipboard,
                                     GdkAtom format,
                                     const guint8 *text,
                                     gsize length,
                                     gpointer data);
```

A function to be called when the results of `gtk_clipboard_request_rich_text()` are received, or when the request fails.

Parameters

clipboard	the GtkClipboard	
format	The format of the rich text	
text	the rich text received, as a UTF-8 encoded string, or `NULL` if retrieving the data failed.	[nullable][type utf8]
length	Length of the text.	
data	the `user_data` supplied to `gtk_clipboard_request_rich_text()`.	[closure]

Since: 2.10

GtkClipboardURIReceivedFunc ()

```
void
(*GtkClipboardURIReceivedFunc) (GtkClipboard *clipboard,
                                gchar **uris,
                                gpointer data);
```

A function to be called when the results of `gtk_clipboard_request_uris()` are received, or when the request fails.

Parameters

clipboard	the GtkClipboard	
uris	the received URIs.	[array zero-terminated=1]
data	the `user_data` supplied to `gtk_clipboard_request_uris()`.	[closure]

Since: 2.14

GtkClipboardGetFunc ()

```
void
(*GtkClipboardGetFunc) (GtkClipboard *clipboard,
                        GtkSelectionData *selection_data,
                        guint info,
                        gpointer user_data_or_owner);
```

A function that will be called to provide the contents of the selection. If multiple types of data were advertised, the requested type can be determined from the `info` parameter or by checking the target field of `selection_data` . If the data could successfully be converted into then it should be stored into the `selection_data` object by calling `gtk_selection_data_set()` (or related functions such as `gtk_selection_data_set_text()`). If no data is set, the requestor will be informed that the attempt to get the data failed.

Parameters

clipboard	the GtkClipboard
selection_data	a GtkSelectionData argument in which the requested data should be stored.
info	the info field corresponding to the requested target from the GtkTargetEntry array passed to `gtk_clipboard_set_with_data()` or `gtk_clipboard_set_with_owner()`.
user_data_or_owner	the `user_data` argument passed to `gtk_clipboard_set_with_data()`, or the `owner` argument passed to `gtk_clipboard_set_with_owner()`

GtkClipboardClearFunc ()

```
void
(*GtkClipboardClearFunc) (GtkClipboard *clipboard,
                          gpointer user_data_or_owner);
```

A function that will be called when the contents of the clipboard are changed or cleared. Once this has called, the `user_data_or_owner` argument will not be used again.

Parameters

clipboard	the GtkClipboard
user_data_or_owner	the `user_data` argument passed to `gtk_clipboard_set_with_data()` , or the `owner` argument passed to `gtk_clipboard_set_with_owner()`

gtk_clipboard_get ()

```
GtkClipboard *
gtk_clipboard_get (GdkAtom selection);
```

Returns the clipboard object for the given selection. See `gtk_clipboard_get_for_display()` for complete details.

Parameters

selection	a GdkAtom which identifies the clipboard to use

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 and, since it is owned by GTK+, must not be freed or unreffed.

[transfer none]

---

gtk_clipboard_get_for_display ()

```
GtkClipboard *
gtk_clipboard_get_for_display (GdkDisplay *display,
                               GdkAtom selection);
```

Returns the clipboard object for the given selection. Cut/copy/paste menu items and keyboard shortcuts should use the default clipboard, returned by passing `GDK_SELECTION_CLIPBOARD` for `selection`. (`GDK_NONE` is supported as a synonym for GDK_SELECTION_CLIPBOARD for backwards compatibility reasons.) The currently-selected object or text should be provided on the clipboard identified by GDK_SELECTION_PRIMARY. Cut/copy/paste menu items conceptually copy the contents of the GDK_SELECTION_PRIMARY clipboard to the default clipboard, i.e. they copy the selection to what the user sees as the clipboard.

(Passing GDK_NONE is the same as using `gdk_atom_intern ("CLIPBOARD", FALSE)`.

See the FreeDesktop Clipboard Specification for a detailed discussion of the “CLIPBOARD” vs. “PRIMARY” selections under the X window system. On Win32 the GDK_SELECTION_PRIMARY clipboard is essentially ignored.)

It’s possible to have arbitrary named clipboards; if you do invent new clipboards, you should prefix the selection name with an underscore (because the ICCCM requires that nonstandard atoms are underscore-prefixed), and namespace it as well. For example, if your application called “Foo” has a special-purpose clipboard, you might call it “_FOO_SPECIAL_CLIPBOARD”.

Parameters

display	the GdkDisplay for which the clipboard is to be retrieved or created.
selection	a GdkAtom which identifies the clipboard to use.

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 and, since it is owned by GTK+, must not be freed or unrefd.

[transfer none]

Since: 2.2

---

gtk_clipboard_get_display ()

```
GdkDisplay *
gtk_clipboard_get_display (GtkClipboard *clipboard);
```

Gets the GdkDisplay associated with `clipboard`

Parameters

clipboard	a GtkClipboard

Returns

the GdkDisplay associated with `clipboard` .

[transfer none]

Since: 2.2

---

gtk_clipboard_get_default ()

```
GtkClipboard *
gtk_clipboard_get_default (GdkDisplay *display);
```

Returns the default clipboard object for use with cut/copy/paste menu items and keyboard shortcuts.

Parameters

display	the GdkDisplay for which the clipboard is to be retrieved.

Returns

the default clipboard object.

[transfer none]

Since: 3.16

---

gtk_clipboard_set_with_data ()

```
gboolean
gtk_clipboard_set_with_data (GtkClipboard *clipboard,
                             const GtkTargetEntry *targets,
                             guint n_targets,
                             GtkClipboardGetFunc get_func,
                             GtkClipboardClearFunc clear_func,
                             gpointer user_data);
```

Virtually sets the contents of the specified clipboard by providing a list of supported formats for the clipboard data and a function to call to get the actual data when it is requested.

[skip]

Parameters

clipboard	a GtkClipboard	
targets	array containing information about the available forms for the clipboard data.	[array length=n_targets]
n_targets	number of elements in `targets`	
get_func	function to call to get the actual clipboard data.	[scope async]
clear_func	when the clipboard contents are set again, this function will be called, and `get_func` will not be subsequently called.	[scope async]
user_data	user data to pass to `get_func` and `clear_func` .	

Returns

`TRUE` if setting the clipboard data succeeded. If setting the clipboard data failed the provided callback functions will be ignored.

---

gtk_clipboard_set_with_owner ()

```
gboolean
gtk_clipboard_set_with_owner (GtkClipboard *clipboard,
                              const GtkTargetEntry *targets,
                              guint n_targets,
                              GtkClipboardGetFunc get_func,
                              GtkClipboardClearFunc clear_func,
                              GObject *owner);
```

Virtually sets the contents of the specified clipboard by providing a list of supported formats for the clipboard data and a function to call to get the actual data when it is requested.

The difference between this function and `gtk_clipboard_set_with_data()` is that instead of an generic `user_data` pointer, a GObject is passed in.

[skip]

Parameters

clipboard	a GtkClipboard	
targets	array containing information about the available forms for the clipboard data.	[array length=n_targets]
n_targets	number of elements in `targets`	
get_func	function to call to get the actual clipboard data.	[scope async]
clear_func	when the clipboard contents are set again, this function will be called, and `get_func` will not be subsequently called.	[scope async]
owner	an object that “owns” the data. This object will be passed to the callbacks when called	

Returns

`TRUE` if setting the clipboard data succeeded. If setting the clipboard data failed the provided callback functions will be ignored.

---

gtk_clipboard_get_owner ()

```
GObject *
gtk_clipboard_get_owner (GtkClipboard *clipboard);
```

If the clipboard contents callbacks were set with `gtk_clipboard_set_with_owner()`, and the `gtk_clipboard_set_with_data()` or `gtk_clipboard_clear()` has not subsequently called, returns the owner set by `gtk_clipboard_set_with_owner()`.

Parameters

clipboard	a GtkClipboard

Returns

the owner of the clipboard, if any; otherwise `NULL`.

[nullable][transfer none]

---

gtk_clipboard_clear ()

```
void
gtk_clipboard_clear (GtkClipboard *clipboard);
```

Clears the contents of the clipboard. Generally this should only be called between the time you call `gtk_clipboard_set_with_owner()` or `gtk_clipboard_set_with_data()`, and when the `clear_func` you supplied is called. Otherwise, the clipboard may be owned by someone else.

Parameters

clipboard	a GtkClipboard

---

gtk_clipboard_set_text ()

```
void
gtk_clipboard_set_text (GtkClipboard *clipboard,
                        const gchar *text,
                        gint len);
```

Sets the contents of the clipboard to the given UTF-8 string. GTK+ will make a copy of the text and take responsibility for responding for requests for the text, and for converting the text into the requested format.

Parameters

clipboard	a GtkClipboard object
text	a UTF-8 string.
len	length of `text` , in bytes, or -1, in which case the length will be determined with `strlen()`.

---

gtk_clipboard_set_image ()

```
void
gtk_clipboard_set_image (GtkClipboard *clipboard,
                         GdkPixbuf *pixbuf);
```

Sets the contents of the clipboard to the given GdkPixbuf. GTK+ will take responsibility for responding for requests for the image, and for converting the image into the requested format.

Parameters

clipboard	a GtkClipboard object
pixbuf	a GdkPixbuf

Since: 2.6

---

gtk_clipboard_request_contents ()

```
void
gtk_clipboard_request_contents (GtkClipboard *clipboard,
                                GdkAtom target,
                                GtkClipboardReceivedFunc callback,
                                gpointer user_data);
```

Requests the contents of clipboard as the given target. When the results of the result are later received the supplied callback will be called.

Parameters

clipboard	a GtkClipboard	
target	an atom representing the form into which the clipboard owner should convert the selection.	
callback	A function to call when the results are received (or the retrieval fails). If the retrieval fails the length field of `selection_data` will be negative.	[scope async]
user_data	user data to pass to `callback`	

---

gtk_clipboard_request_text ()

```
void
gtk_clipboard_request_text (GtkClipboard *clipboard,
                            GtkClipboardTextReceivedFunc callback,
                            gpointer user_data);
```

Requests the contents of the clipboard as text. When the text is later received, it will be converted to UTF-8 if necessary, and `callback` will be called.

The `text` parameter to `callback` will contain the resulting text if the request succeeded, or `NULL` if it failed. This could happen for various reasons, in particular if the clipboard was empty or if the contents of the clipboard could not be converted into text form.

Parameters

clipboard	a GtkClipboard	
callback	a function to call when the text is received, or the retrieval fails. (It will always be called one way or the other.).	[scope async]
user_data	user data to pass to `callback` .	

---

gtk_clipboard_request_image ()

```
void
gtk_clipboard_request_image (GtkClipboard *clipboard,
                             GtkClipboardImageReceivedFunc callback,
                             gpointer user_data);
```

Requests the contents of the clipboard as image. When the image is later received, it will be converted to a GdkPixbuf, and `callback` will be called.

The `pixbuf` parameter to `callback` will contain the resulting GdkPixbuf if the request succeeded, or `NULL` if it failed. This could happen for various reasons, in particular if the clipboard was empty or if the contents of the clipboard could not be converted into an image.

Parameters

clipboard	a GtkClipboard	
callback	a function to call when the image is received, or the retrieval fails. (It will always be	[scope async]

	called one way or the other.).
user_data	user data to pass to `callback` .

Since: 2.6

---

gtk_clipboard_request_targets ()

```
void
gtk_clipboard_request_targets (GtkClipboard *clipboard,
                               GtkClipboardTargetsReceivedFunc callback,
                               gpointer user_data);
```

Requests the contents of the clipboard as list of supported targets. When the list is later received, `callback` will be called.

The `targets` parameter to `callback` will contain the resulting targets if the request succeeded, or `NULL` if it failed.

Parameters

clipboard	a GtkClipboard	
callback	a function to call when the targets are received, or the retrieval fails. (It will always be called one way or the other.).	[scope async]
user_data	user data to pass to `callback` .	

Since: 2.4

---

gtk_clipboard_request_rich_text ()

```
void
gtk_clipboard_request_rich_text (GtkClipboard *clipboard,
                                 GtkTextBuffer *buffer,
                                 GtkClipboardRichTextReceivedFunc callback,
                                 gpointer user_data);
```

Requests the contents of the clipboard as rich text. When the rich text is later received, `callback` will be called.

The `text` parameter to `callback` will contain the resulting rich text if the request succeeded, or `NULL` if it failed. The `length` parameter will contain `text` 's length. This function can fail for various reasons, in particular if the clipboard was empty or if the contents of the clipboard could not be converted into rich text form.

Parameters

clipboard	a GtkClipboard	
buffer	a GtkTextBuffer	
callback	a function to call when the text is received, or the retrieval fails. (It will always be called one way or the other.).	[scope async]
user_data	user data to pass to `callback` .	

Since: 2.10

---

gtk_clipboard_request_uris ()

```
void
gtk_clipboard_request_uris (GtkClipboard *clipboard,
                            GtkClipboardURIReceivedFunc callback,
                            gpointer user_data);
```

Requests the contents of the clipboard as URIs. When the URIs are later received `callback` will be called.

The `uris` parameter to `callback` will contain the resulting array of URIs if the request succeeded, or `NULL` if it failed. This could happen for various reasons, in particular if the clipboard was empty or if the contents of the clipboard could not be converted into URI form.

Parameters

clipboard	a GtkClipboard	
callback	a function to call when the URIs are received, or the retrieval fails. (It will always be called one way or the other.).	[scope async]
user_data	user data to pass to `callback` .	

Since: 2.14

---

gtk_clipboard_wait_for_contents ()

```
GtkSelectionData *
gtk_clipboard_wait_for_contents (GtkClipboard *clipboard,
                                 GdkAtom target);
```

Requests the contents of the clipboard using the given target. This function waits for the data to be received using the main loop, so events, timeouts, etc, may be dispatched during the wait.

Parameters

clipboard	a GtkClipboard
target	an atom representing the form into which the clipboard owner should convert the selection.

Returns

a newly-allocated GtkSelectionData object or `NULL` if retrieving the given target failed. If non-`NULL`, this value must be freed with `gtk_selection_data_free()` when you are finished with it.

[nullable]

---

gtk_clipboard_wait_for_text ()

```
gchar *
gtk_clipboard_wait_for_text (GtkClipboard *clipboard);
```

Requests the contents of the clipboard as text and converts the result to UTF-8 if necessary. This function waits for the data to be received using the main loop, so events, timeouts, etc, may be dispatched during the wait.

Parameters

clipboard	a GtkClipboard

Returns

a newly-allocated UTF-8 string which must be freed with `g_free()`, or `NULL` if retrieving the selection data failed. (This could happen for various reasons, in particular if the clipboard was empty or if the contents of the clipboard could not be converted into text form.).

[nullable]

---

gtk_clipboard_wait_for_image ()

```
GdkPixbuf *
gtk_clipboard_wait_for_image (GtkClipboard *clipboard);
```

Requests the contents of the clipboard as image and converts the result to a GdkPixbuf. This function waits for the data to be received using the main loop, so events, timeouts, etc, may be dispatched during the wait.

Parameters

clipboard	a GtkClipboard

Returns

a newly-allocated GdkPixbuf object which must be disposed with `g_object_unref()`, or `NULL` if retrieving the selection data failed. (This could happen for various reasons, in particular if the clipboard was empty or if the contents of the clipboard could not be converted into an image.).

[nullable][transfer full]

Since: 2.6

---

gtk_clipboard_wait_for_rich_text ()

```
guint8 *
gtk_clipboard_wait_for_rich_text (GtkClipboard *clipboard,
                                  GtkTextBuffer *buffer,
                                  GdkAtom *format,
                                  gsize *length);
```

Requests the contents of the clipboard as rich text. This function waits for the data to be received using the main loop, so events, timeouts, etc, may be dispatched during the wait.

Parameters

clipboard	a GtkClipboard	
buffer	a GtkTextBuffer	
format	return location for the format of the returned data.	[out]
length	return location for the length of the returned data.	[out]

Returns

a newly-allocated binary block of data which must be freed with `g_free()`, or `NULL` if retrieving the selection data failed. (This could happen for various reasons, in particular if the clipboard was empty or if the contents of the clipboard could not be converted into text form.).

[nullable][array length=length][transfer full]

Since: 2.10

---

gtk_clipboard_wait_for_uris ()

```
gchar **
gtk_clipboard_wait_for_uris (GtkClipboard *clipboard);
```

Requests the contents of the clipboard as URIs. This function waits for the data to be received using the main loop, so events, timeouts, etc, may be dispatched during the wait.

Parameters

clipboard	a GtkClipboard

Returns

a newly-allocated `NULL`-terminated array of strings which must be freed with `g_strfreev()`, or `NULL` if retrieving the selection data failed. (This could happen for various reasons, in particular if the clipboard was empty or if the contents of the clipboard could not be converted into URI form.).

[nullable][array zero-terminated=1][element-type utf8][transfer full]

Since: 2.14

---

gtk_clipboard_wait_is_text_available ()

```
gboolean
gtk_clipboard_wait_is_text_available (GtkClipboard *clipboard);
```

Test to see if there is text available to be pasted This is done by requesting the TARGETS atom and checking if it contains any of the supported text targets. This function waits for the data to be received using the main loop,

so events, timeouts, etc, may be dispatched during the wait.

This function is a little faster than calling `gtk_clipboard_wait_for_text()` since it doesn't need to retrieve the actual text.

Parameters

clipboard	a GtkClipboard

Returns

`TRUE` is there is text available, `FALSE` otherwise.

---

gtk_clipboard_wait_is_image_available ()

```
gboolean
gtk_clipboard_wait_is_image_available (GtkClipboard *clipboard);
```

Test to see if there is an image available to be pasted This is done by requesting the TARGETS atom and checking if it contains any of the supported image targets. This function waits for the data to be received using the main loop, so events, timeouts, etc, may be dispatched during the wait.

This function is a little faster than calling `gtk_clipboard_wait_for_image()` since it doesn't need to retrieve the actual image data.

Parameters

clipboard	a GtkClipboard

Returns

`TRUE` is there is an image available, `FALSE` otherwise.

Since: 2.6

---

gtk_clipboard_wait_is_rich_text_available ()

```
gboolean
gtk_clipboard_wait_is_rich_text_available
                                (GtkClipboard *clipboard,
                                 GtkTextBuffer *buffer);
```

Test to see if there is rich text available to be pasted This is done by requesting the TARGETS atom and checking if it contains any of the supported rich text targets. This function waits for the data to be received using the main loop, so events, timeouts, etc, may be dispatched during the wait.

This function is a little faster than calling `gtk_clipboard_wait_for_rich_text()` since it doesn't need to retrieve the actual text.

Parameters

clipboard	a GtkClipboard
buffer	a GtkTextBuffer

Returns

`TRUE` is there is rich text available, `FALSE` otherwise.

Since: 2.10

---

gtk_clipboard_wait_is_uris_available ()

```
gboolean
gtk_clipboard_wait_is_uris_available (GtkClipboard *clipboard);
```

Test to see if there is a list of URIs available to be pasted This is done by requesting the TARGETS atom and checking if it contains the URI targets. This function waits for the data to be received using the main loop, so events, timeouts, etc, may be dispatched during the wait.

This function is a little faster than calling `gtk_clipboard_wait_for_uris()` since it doesn't need to retrieve the actual URI data.

Parameters

clipboard	a GtkClipboard

Returns

`TRUE` is there is an URI list available, `FALSE` otherwise.

Since: 2.14

---

gtk_clipboard_wait_for_targets ()

```
gboolean
gtk_clipboard_wait_for_targets (GtkClipboard *clipboard,
                                GdkAtom **targets,
                                gint *n_targets);
```

Returns a list of targets that are present on the clipboard, or `NULL` if there aren't any targets available. The returned list must be freed with `g_free()`. This function waits for the data to be received using the main loop, so events, timeouts, etc, may be dispatched during the wait.

Parameters

clipboard	a GtkClipboard	
targets	location to store an array of targets. The result stored here must be freed with `g_free()`.	[out][array length=n_targets] [transfer container]
n_targets	location to store number of items in `targets`.	[out]

Returns

`TRUE` if any targets are present on the clipboard, otherwise `FALSE`.

Since: 2.4

---

gtk_clipboard_wait_is_target_available ()

```
gboolean
gtk_clipboard_wait_is_target_available
                                (GtkClipboard *clipboard,
                                 GdkAtom target);
```

Checks if a clipboard supports pasting data of a given type. This function can be used to determine if a “Paste” menu item should be insensitive or not.

If you want to see if there’s text available on the clipboard, use `gtk_clipboard_wait_is_text_available()` instead.

Parameters

clipboard	a GtkClipboard
target	A GdkAtom indicating which target to look for.

Returns

`TRUE` if the target is available, `FALSE` otherwise.

Since: 2.6

---

gtk_clipboard_set_can_store ()

```
void
gtk_clipboard_set_can_store (GtkClipboard *clipboard,
                             const GtkTargetEntry *targets,
                             gint n_targets);
```

Hints that the clipboard data should be stored somewhere when the application exits or when `gtk_clipboard_store()` is called.

This value is reset when the clipboard owner changes. Where the clipboard data is stored is platform dependent,

see `gdk_display_store_clipboard()` for more information.

Parameters

clipboard	a GtkClipboard	
targets	array containing information about which forms should be stored or `NULL` to indicate that all forms should be stored.	[allow-none][array length=n_targets]
n_targets	number of elements in `targets`	

Since: 2.6

---

gtk_clipboard_store ()

```
void
gtk_clipboard_store (GtkClipboard *clipboard);
```

Stores the current clipboard data somewhere so that it will stay around after the application has quit.

Parameters

clipboard	a GtkClipboard

Since: 2.6

---

gtk_clipboard_get_selection ()

```
GdkAtom
gtk_clipboard_get_selection (GtkClipboard *clipboard);
```

Gets the selection that this clipboard is for.

Parameters

clipboard	a GtkClipboard

Returns

the selection

Since: 3.22

Types and Values

GtkClipboard

```
typedef struct _GtkClipboard GtkClipboard;
```

Signal Details

The "owner-change" signal

```
void
user_function (GtkClipboard *clipboard,
               GdkEvent     *event,
               gpointer      user_data)
```

The ::owner-change signal is emitted when GTK+ receives an event that indicates that the ownership of the selection associated with `clipboard` has changed.

Parameters

clipboard	the GtkClipboard on which the signal is emitted	
event	the `GdkEventOwnerChange` event.	[type Gdk.EventOwnerChange]
user_data	user data set when the signal handler was connected.	

Flags: Run First

Since: 2.6

See Also

GtkSelectionData

Drag and Drop

Drag and Drop — Functions for controlling drag and drop handling

Functions

void	gtk_drag_dest_set ()
void	gtk_drag_dest_set_proxy ()
void	gtk_drag_dest_unset ()
GdkAtom	gtk_drag_dest_find_target ()
GtkTargetList *	gtk_drag_dest_get_target_list ()
void	gtk_drag_dest_set_target_list ()
void	gtk_drag_dest_add_text_targets ()
void	gtk_drag_dest_add_image_targets ()
void	gtk_drag_dest_add_uri_targets ()
void	gtk_drag_dest_set_track_motion ()

gboolean	gtk_drag_dest_get_track_motion ()
void	gtk_drag_finish ()
void	gtk_drag_get_data ()
GtkWidget *	gtk_drag_get_source_widget ()
void	gtk_drag_highlight ()
void	gtk_drag_unhighlight ()
GdkDragContext *	gtk_drag_begin ()
GdkDragContext *	gtk_drag_begin_with_coordinates ()
void	gtk_drag_cancel ()
void	gtk_drag_set_icon_widget ()
void	gtk_drag_set_icon_pixbuf ()
void	gtk_drag_set_icon_stock ()
void	gtk_drag_set_icon_surface ()
void	gtk_drag_set_icon_name ()
void	gtk_drag_set_icon_gicon ()
void	gtk_drag_set_icon_default ()
gboolean	gtk_drag_check_threshold ()
void	gtk_drag_source_set ()
void	gtk_drag_source_set_icon_pixbuf ()
void	gtk_drag_source_set_icon_stock ()
void	gtk_drag_source_set_icon_name ()
void	gtk_drag_source_set_icon_gicon ()
void	gtk_drag_source_unset ()
void	gtk_drag_source_set_target_list ()
GtkTargetList *	gtk_drag_source_get_target_list ()
void	gtk_drag_source_add_text_targets ()
void	gtk_drag_source_add_image_targets ()
void	gtk_drag_source_add_uri_targets ()

Types and Values

enum	GtkDestDefaults
enum	GtkTargetFlags
enum	GtkDragResult

Includes

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

Description

GTK+ has a rich set of functions for doing inter-process communication via the drag-and-drop metaphor.

As well as the functions listed here, applications may need to use some facilities provided for Selections. Also, the Drag and Drop API makes use of signals in the GtkWidget class.

Functions

gtk_drag_dest_set ()

```
void
gtk_drag_dest_set (GtkWidget *widget,
                   GtkDestDefaults flags,
                   const GtkTargetEntry *targets,
                   gint n_targets,
                   GdkDragAction actions);
```

Sets a widget as a potential drop destination, and adds default behaviors.

The default behaviors listed in `flags` have an effect similar to installing default handlers for the widget's drag-and-drop signals ("drag-motion", "drag-drop", ...). They all exist for convenience. When passing GTK_DEST_DEFAULT_ALL for instance it is sufficient to connect to the widget's "drag-data-received" signal to get primitive, but consistent drag-and-drop support.

Things become more complicated when you try to preview the dragged data, as described in the documentation for "drag-motion". The default behaviors described by `flags` make some assumptions, that can conflict with your own signal handlers. For instance GTK_DEST_DEFAULT_DROP causes invokations of `gdk_drag_status()` in the context of "drag-motion", and invokations of `gtk_drag_finish()` in "drag-data-received". Especially the later is dramatic, when your own "drag-motion" handler calls `gtk_drag_get_data()` to inspect the dragged data.

There's no way to set a default action here, you can use the "drag-motion" callback for that. Here's an example which selects the action to use depending on whether the control key is pressed or not:

```
1  static void
2  drag_motion (GtkWidget *widget,
3               GdkDragContext *context,
4               gint x,
5               gint y,
6               guint time)
7  {
8    GdkModifierType mask;
9
10   gdk_window_get_pointer (gtk_widget_get_window (widget),
11                           NULL, NULL, &mask);
12   if (mask & GDK_CONTROL_MASK)
13     gdk_drag_status (context, GDK_ACTION_COPY, time);
14   else
15     gdk_drag_status (context, GDK_ACTION_MOVE, time);
16 }
```

[method]

Parameters

widget	a GtkWidget	
flags	which types of default drag behavior to use	
targets	a pointer to an array of GtkTargetEntrys indicating the drop types that this `widget` will accept, or `NULL`. Later you can access the list with `gtk_drag_dest_get_target_list()` and `gtk_drag_dest_find_target()`.	[allow-none][array length=n_targets]

n_targets	the number of entries in `targets`
actions	a bitmask of possible actions for a drop onto this `widget` .

---

gtk_drag_dest_set_proxy ()

```
void
gtk_drag_dest_set_proxy (GtkWidget *widget,
                         GdkWindow *proxy_window,
                         GdkDragProtocol protocol,
                         gboolean use_coordinates);
```

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

Sets this widget as a proxy for drops to another window.

[method]

Parameters

widget	a GtkWidget
proxy_window	the window to which to forward drag events
protocol	the drag protocol which the `proxy_window` accepts (You can use `gdk_drag_get_protocol()` to determine this)
use_coordinates	If `TRUE`, send the same coordinates to the destination, because it is an embedded subwindow.

---

gtk_drag_dest_unset ()

```
void
gtk_drag_dest_unset (GtkWidget *widget);
```

Clears information about a drop destination set with `gtk_drag_dest_set()`. The widget will no longer receive notification of drags.

[method]

Parameters

widget	a GtkWidget

---

gtk_drag_dest_find_target ()

```
GdkAtom
gtk_drag_dest_find_target (GtkWidget *widget,
                           GdkDragContext *context,
                           GtkTargetList *target_list);
```

Looks for a match between the supported targets of `context` and the `dest_target_list` , returning the first matching target, otherwise returning `GDK_NONE`. `dest_target_list` should usually be the return value from `gtk_drag_dest_get_target_list()`, but some widgets may have different valid targets for different parts of the widget; in that case, they will have to implement a drag_motion handler that passes the correct target list to this function.

[method]

Parameters

widget	drag destination widget	
context	drag context	
target_list	list of droppable targets, or `NULL` to use gtk_drag_dest_get_target_list (`widget`).	[allow-none]

Returns

first target that the source offers and the dest can accept, or `GDK_NONE`.

[transfer none]

---

gtk_drag_dest_get_target_list ()

```
GtkTargetList *
gtk_drag_dest_get_target_list (GtkWidget *widget);
```

Returns the list of targets this widget can accept from drag-and-drop.

[method]

Parameters

widget	a GtkWidget

Returns

the GtkTargetList, or `NULL` if none.

[nullable][transfer none]

---

gtk_drag_dest_set_target_list ()

```
void
gtk_drag_dest_set_target_list (GtkWidget *widget,
                               GtkTargetList *target_list);
```

Sets the target types that this widget can accept from drag-and-drop. The widget must first be made into a drag destination with `gtk_drag_dest_set()`.

[method]

Parameters

widget	a GtkWidget that's a drag destination	
target_list	list of droppable targets, or `NULL` for none.	[allow-none]

---

gtk_drag_dest_add_text_targets ()

```
void
gtk_drag_dest_add_text_targets (GtkWidget *widget);
```

Add the text targets supported by GtkSelectionData to the target list of the drag destination. The targets are added with `info` = 0. If you need another value, use `gtk_target_list_add_text_targets()` and `gtk_drag_dest_set_target_list()`.

[method]

Parameters

widget	a GtkWidget that's a drag destination

Since: 2.6

---

gtk_drag_dest_add_image_targets ()

```
void
gtk_drag_dest_add_image_targets (GtkWidget *widget);
```

Add the image targets supported by GtkSelectionData to the target list of the drag destination. The targets are added with `info` = 0. If you need another value, use `gtk_target_list_add_image_targets()` and `gtk_drag_dest_set_target_list()`.

[method]

Parameters

widget	a GtkWidget that's a drag destination

Since: 2.6

---

gtk_drag_dest_add_uri_targets ()

```
void
gtk_drag_dest_add_uri_targets (GtkWidget *widget);
```

Add the URI targets supported by GtkSelectionData to the target list of the drag destination. The targets are added with `info` = 0. If you need another value, use `gtk_target_list_add_uri_targets()` and `gtk_drag_dest_set_target_list()`.

[method]

Parameters

widget	a GtkWidget that's a drag destination

Since: 2.6

---

gtk_drag_dest_set_track_motion ()

```
void
gtk_drag_dest_set_track_motion (GtkWidget *widget,
                                gboolean track_motion);
```

Tells the widget to emit "drag-motion" and "drag-leave" events regardless of the targets and the `GTK_DEST_DEFAULT_MOTION` flag.

This may be used when a widget wants to do generic actions regardless of the targets that the source offers.

[method]

Parameters

widget	a GtkWidget that's a drag destination
track_motion	whether to accept all targets

Since: 2.10

---

gtk_drag_dest_get_track_motion ()

```
gboolean
gtk_drag_dest_get_track_motion (GtkWidget *widget);
```

Returns whether the widget has been configured to always emit "drag-motion" signals.

[method]

Parameters

widget	a GtkWidget that's a drag destination

Returns

`TRUE` if the widget always emits "drag-motion" events

Since: 2.10

---

gtk_drag_finish ()

```
void
gtk_drag_finish (GdkDragContext *context,
                 gboolean success,
                 gboolean del,
                 guint32 time_);
```

Informs the drag source that the drop is finished, and that the data of the drag will no longer be required.

[method]

Parameters

context	the drag context
success	a flag indicating whether the drop was successful
del	a flag indicating whether the source should delete the original data. (This should be `TRUE` for a move)
time_	the timestamp from the "drag-drop" signal

---

gtk_drag_get_data ()

```
void
gtk_drag_get_data (GtkWidget *widget,
                   GdkDragContext *context,
                   GdkAtom target,
                   guint32 time_);
```

Gets the data associated with a drag. When the data is received or the retrieval fails, GTK+ will emit a "drag-data-received" signal. Failure of the retrieval is indicated by the length field of the `selection_data` signal parameter being negative. However, when `gtk_drag_get_data()` is called implicitely because the `GTK_DEST_DEFAULT_DROP` was set, then the widget will not receive notification of failed drops.

[method]

Parameters

widget	the widget that will receive the "drag-data-received" signal
context	the drag context
target	the target (form of the data) to retrieve
time_	a timestamp for retrieving the data. This will generally be the time received in a "drag-motion" or "drag-drop" signal

---

gtk_drag_get_source_widget ()

```
GtkWidget *
gtk_drag_get_source_widget (GdkDragContext *context);
```

Determines the source widget for a drag.

[method]

Parameters

context	a (destination side) drag context

Returns

if the drag is occurring within a single application, a pointer to the source widget. Otherwise, `NULL`.

[nullable][transfer none]

---

gtk_drag_highlight ()

```
void
gtk_drag_highlight (GtkWidget *widget);
```

Highlights a widget as a currently hovered drop target. To end the highlight, call `gtk_drag_unhighlight()`. GTK+ calls this automatically if `GTK_DEST_DEFAULT_HIGHLIGHT` is set.

[method]

Parameters

widget	a widget to highlight

---

gtk_drag_unhighlight ()

```
void
gtk_drag_unhighlight (GtkWidget *widget);
```

Removes a highlight set by `gtk_drag_highlight()` from a widget.

[method]

Parameters

widget	a widget to remove the highlight from

---

gtk_drag_begin ()

```
GdkDragContext *
gtk_drag_begin (GtkWidget *widget,
                GtkTargetList *targets,
                GdkDragAction actions,
                gint button,
                GdkEvent *event);
```

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

Use `gtk_drag_begin_with_coordinates()` instead

This function is equivalent to `gtk_drag_begin_with_coordinates()`, passing -1, -1 as coordinates.

[method]

Parameters

widget	the source widget	
targets	The targets (data formats) in which the source can provide the data	
actions	A bitmask of the allowed drag actions for this drag	
button	The button the user clicked to start the drag	
event	The event that triggered the start of the drag, or `NULL` if none can be obtained.	[nullable]

Returns

the context for this drag.

[transfer none]

---

gtk_drag_begin_with_coordinates ()

```
GdkDragContext *
gtk_drag_begin_with_coordinates (GtkWidget *widget,
                                 GtkTargetList *targets,
                                 GdkDragAction actions,
                                 gint button,
                                 GdkEvent *event,
                                 gint x,
                                 gint y);
```

Initiates a drag on the source side. The function only needs to be used when the application is starting drags itself, and is not needed when `gtk_drag_source_set()` is used.

The `event` is used to retrieve the timestamp that will be used internally to grab the pointer. If `event` is `NULL`, then `GDK_CURRENT_TIME` will be used. However, you should try to pass a real event in all cases, since that can be used to get information about the drag.

Generally there are three cases when you want to start a drag by hand by calling this function:

1. During a "button-press-event" handler, if you want to start a drag immediately when the user presses the mouse button. Pass the `event` that you have in your "button-press-event" handler.

2. During a “motion-notify-event” handler, if you want to start a drag when the mouse moves past a certain threshold distance after a button-press. Pass the `event` that you have in your “motion-notify-event” handler.

3. During a timeout handler, if you want to start a drag after the mouse button is held down for some time. Try to save the last event that you got from the mouse, using `gdk_event_copy()`, and pass it to this function (remember to free the event with `gdk_event_free()` when you are done). If you really cannot pass a real event, pass `NULL` instead.

[method]

Parameters

widget	the source widget	
targets	The targets (data formats) in which the source can provide the data	
actions	A bitmask of the allowed drag actions for this drag	
button	The button the user clicked to start the drag	
event	The event that triggered the start of the drag, or `NULL` if none can be obtained.	[nullable]
x	The initial x coordinate to start dragging from, in the coordinate space of `widget` . If -1 is passed, the coordinates are retrieved from `event` or the current pointer position	
y	The initial y coordinate to start dragging from, in the coordinate space of `widget` . If -1 is passed, the coordinates are retrieved from `event` or the current pointer position	

Returns

the context for this drag.

[transfer none]

Since: 3.10

---

gtk_drag_cancel ()

```
void
gtk_drag_cancel (GdkDragContext *context);
```

Cancels an ongoing drag operation on the source side.

If you want to be able to cancel a drag operation in this way, you need to keep a pointer to the drag context, either from an explicit call to `gtk_drag_begin_with_coordinates()`, or by connecting to “drag-begin”.

If `context` does not refer to an ongoing drag operation, this function does nothing.

If a drag is cancelled in this way, the `result` argument of “drag-failed” is set to `GTK_DRAG_RESULT_ERROR` .

[method]

Parameters

context	a GdkDragContext, as e.g. returned by

gtk_drag_begin_with_coordinates()

Since: 3.16

---

gtk_drag_set_icon_widget ()

```
void
gtk_drag_set_icon_widget (GdkDragContext *context,
                          GtkWidget *widget,
                          gint hot_x,
                          gint hot_y);
```

Changes the icon for drag operation to a given widget. GTK+ will not destroy the widget, so if you don’t want it to persist, you should connect to the “drag-end” signal and destroy it yourself.

[method]

Parameters

context	the context for a drag. (This must be called with a context for the source side of a drag)
widget	a widget to use as an icon
hot_x	the X offset within `widget` of the hotspot
hot_y	the Y offset within `widget` of the hotspot

---

gtk_drag_set_icon_pixbuf ()

```
void
gtk_drag_set_icon_pixbuf (GdkDragContext *context,
                          GdkPixbuf *pixbuf,
                          gint hot_x,
                          gint hot_y);
```

Sets `pixbuf` as the icon for a given drag.

Parameters

context	the context for a drag (This must be called with a context for the source side of a drag)
pixbuf	the GdkPixbuf to use as the drag icon
hot_x	the X offset within `widget` of the hotspot
hot_y	the Y offset within `widget` of the hotspot

---

gtk_drag_set_icon_stock ()

```
void
gtk_drag_set_icon_stock (GdkDragContext *context,
                         const gchar *stock_id,
                         gint hot_x,
                         gint hot_y);
```

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

Use gtk_drag_set_icon_name() instead.

Sets the icon for a given drag from a stock ID.

Parameters

context	the context for a drag (This must be called with a context for the source side of a drag)
stock_id	the ID of the stock icon to use for the drag
hot_x	the X offset within the icon of the hotspot
hot_y	the Y offset within the icon of the hotspot

---

gtk_drag_set_icon_surface ()

```
void
gtk_drag_set_icon_surface (GdkDragContext *context,
                           cairo_surface_t *surface);
```

Sets surface as the icon for a given drag. GTK+ retains references for the arguments, and will release them when they are no longer needed.

To position the surface relative to the mouse, use cairo_surface_set_device_offset() on surface . The mouse cursor will be positioned at the (0,0) coordinate of the surface.

Parameters

context	the context for a drag (This must be called with a context for the source side of a drag)
surface	the surface to use as icon

---

gtk_drag_set_icon_name ()

```
void
gtk_drag_set_icon_name (GdkDragContext *context,
                        const gchar *icon_name,
                        gint hot_x,
                        gint hot_y);
```

Sets the icon for a given drag from a named themed icon. See the docs for GtkIconTheme for more details. Note that the size of the icon depends on the icon theme (the icon is loaded at the symbolic size GTK_ICON_SIZE_DND), thus hot_x and hot_y have to be used with care.

Parameters

context	the context for a drag (This must be called with a context for the source side of a drag)
icon_name	name of icon to use

hot_x	the X offset of the hotspot within the icon
hot_y	the Y offset of the hotspot within the icon

Since: 2.8

---

gtk_drag_set_icon_gicon ()

```
void
gtk_drag_set_icon_gicon (GdkDragContext *context,
                         GIcon *icon,
                         gint hot_x,
                         gint hot_y);
```

Sets the icon for a given drag from the given `icon` . See the documentation for `gtk_drag_set_icon_name()` for more details about using icons in drag and drop.

Parameters

context	the context for a drag (This must be called with a context for the source side of a drag)
icon	a GIcon
hot_x	the X offset of the hotspot within the icon
hot_y	the Y offset of the hotspot within the icon

Since: 3.2

---

gtk_drag_set_icon_default ()

```
void
gtk_drag_set_icon_default (GdkDragContext *context);
```

Sets the icon for a particular drag to the default icon.

[method]

Parameters

context	the context for a drag (This must be called with a context for the source side of a drag)

---

gtk_drag_check_threshold ()

```
gboolean
gtk_drag_check_threshold (GtkWidget *widget,
                          gint start_x,
                          gint start_y,
                          gint current_x,
                          gint current_y);
```

Checks to see if a mouse drag starting at (`start_x` , `start_y`) and ending at (`current_x` , `current_y`) has

passed the GTK+ drag threshold, and thus should trigger the beginning of a drag-and-drop operation.

[method]

Parameters

widget	a GtkWidget
start_x	X coordinate of start of drag
start_y	Y coordinate of start of drag
current_x	current X coordinate
current_y	current Y coordinate

Returns

`TRUE` if the drag threshold has been passed.

---

gtk_drag_source_set ()

```
void
gtk_drag_source_set (GtkWidget *widget,
                     GdkModifierType start_button_mask,
                     const GtkTargetEntry *targets,
                     gint n_targets,
                     GdkDragAction actions);
```

Sets up a widget so that GTK+ will start a drag operation when the user clicks and drags on the widget. The widget must have a window.

[method]

Parameters

widget	a GtkWidget	
start_button_mask	the bitmask of buttons that can start the drag	
targets	the table of targets that the drag will support, may be `NULL`.	[allow-none][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_drag_source_set_icon_pixbuf ()

```
void
gtk_drag_source_set_icon_pixbuf (GtkWidget *widget,
                                 GdkPixbuf *pixbuf);
```

Sets the icon that will be used for drags from a particular widget from a GdkPixbuf. GTK+ retains a reference for `pixbuf` and will release it when it is no longer needed.

[method]

Parameters

widget	a GtkWidget
pixbuf	the GdkPixbuf for the drag icon

---

gtk_drag_source_set_icon_stock ()

```
void
gtk_drag_source_set_icon_stock (GtkWidget *widget,
                                const gchar *stock_id);
```

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

Use `gtk_drag_source_set_icon_name()` instead.

Sets the icon that will be used for drags from a particular source to a stock icon.

[method]

Parameters

widget	a GtkWidget
stock_id	the ID of the stock icon to use

---

gtk_drag_source_set_icon_name ()

```
void
gtk_drag_source_set_icon_name (GtkWidget *widget,
                               const gchar *icon_name);
```

Sets the icon that will be used for drags from a particular source to a themed icon. See the docs for GtkIconTheme for more details.

[method]

Parameters

widget	a GtkWidget
icon_name	name of icon to use

Since: 2.8

---

gtk_drag_source_set_icon_gicon ()

```
void
gtk_drag_source_set_icon_gicon (GtkWidget *widget,
                                GIcon *icon);
```

Sets the icon that will be used for drags from a particular source to `icon` . See the docs for GtkIconTheme for more details.

[method]

Parameters

widget	a GtkWidget
icon	A GIcon

Since: 3.2

---

gtk_drag_source_unset ()

```
void
gtk_drag_source_unset (GtkWidget *widget);
```

Undoes the effects of `gtk_drag_source_set()`.

[method]

Parameters

widget	a GtkWidget

---

gtk_drag_source_set_target_list ()

```
void
gtk_drag_source_set_target_list (GtkWidget *widget,
                                 GtkTargetList *target_list);
```

Changes the target types that this widget offers for drag-and-drop. The widget must first be made into a drag source with `gtk_drag_source_set()`.

[method]

Parameters

widget	a GtkWidget that's a drag source	
target_list	list of draggable targets, or `NULL` for none.	[allow-none]

Since: 2.4

---

gtk_drag_source_get_target_list ()

```
GtkTargetList *
gtk_drag_source_get_target_list (GtkWidget *widget);
```

Gets the list of targets this widget can provide for drag-and-drop.

[method]

Parameters

widget	a GtkWidget

Returns

the GtkTargetList, or `NULL` if none.

[nullable][transfer none]

Since: 2.4

---

gtk_drag_source_add_text_targets ()

```
void
gtk_drag_source_add_text_targets (GtkWidget *widget);
```

Add the text targets supported by GtkSelectionData to the target list of the drag source. The targets are added with `info` = 0. If you need another value, use `gtk_target_list_add_text_targets()` and `gtk_drag_source_set_target_list()`.

[method]

Parameters

widget	a GtkWidget that's is a drag source

Since: 2.6

---

gtk_drag_source_add_image_targets ()

```
void
gtk_drag_source_add_image_targets (GtkWidget *widget);
```

Add the writable image targets supported by GtkSelectionData to the target list of the drag source. The targets are added with `info` = 0. If you need another value, use `gtk_target_list_add_image_targets()` and `gtk_drag_source_set_target_list()`.

[method]

Parameters

widget	a GtkWidget that's is a drag source

Since: 2.6

---

gtk_drag_source_add_uri_targets ()

```
void
gtk_drag_source_add_uri_targets (GtkWidget *widget);
```

Add the URI targets supported by GtkSelectionData to the target list of the drag source. The targets are added with `info` = 0. If you need another value, use `gtk_target_list_add_uri_targets()` and `gtk_drag_source_set_target_list()`.

[method]

Parameters

widget	a GtkWidget that's is a drag source

Since: 2.6

Types and Values

enum GtkDestDefaults

The GtkDestDefaults enumeration specifies the various types of action that will be taken on behalf of the user for a drag destination site.

Members

GTK_DEST_DEFAULT_MOTION	If set for a widget, GTK+, during a drag over this widget will check if the drag matches this widget's list of possible targets and actions. GTK+ will then call `gdk_drag_status()` as appropriate.
GTK_DEST_DEFAULT_HIGHLIGHT	If set for a widget, GTK+ will draw a highlight on this widget as long as a drag is over this widget and the widget drag format and action are acceptable.
GTK_DEST_DEFAULT_DROP	If set for a widget, when a drop occurs, GTK+ will will check if the drag matches this widget's list of possible targets and actions. If so, GTK+ will call `gtk_drag_get_data()` on behalf of the widget. Whether or not the drop is successful, GTK+ will call `gtk_drag_finish()`. If the action was a move, then if the drag was successful, then `TRUE` will be passed for the `delete` parameter to `gtk_drag_finish()`.
GTK_DEST_DEFAULT_ALL	If set, specifies that all default actions should be taken.

---

enum GtkTargetFlags

The GtkTargetFlags enumeration is used to specify constraints on a GtkTargetEntry.

Members

GTK_TARGET_SAME_APP	If this is set, the target will only be selected for drags within a single application.
GTK_TARGET_SAME_WIDGET	If this is set, the target will only be selected for drags within a single widget.
GTK_TARGET_OTHER_APP	If this is set, the target will not be selected for drags within a single application.
GTK_TARGET_OTHER_WIDGET	If this is set, the target will not be selected for drags withing a single widget.

---

enum GtkDragResult

Gives an indication why a drag operation failed. The value can by obtained by connecting to the “drag-failed” signal.

Members

GTK_DRAG_RESULT_SUCCESS	The drag operation was successful.
GTK_DRAG_RESULT_NO_TARGET	No suitable drag target.
GTK_DRAG_RESULT_USER_CANCELLED	The user cancelled the drag operation.
GTK_DRAG_RESULT_TIMEOUT_EXPIRED	The drag operation timed out.
GTK_DRAG_RESULT_GRAB_BROKEN	The pointer or keyboard grab used for the drag operation was broken.
GTK_DRAG_RESULT_ERROR	The drag operation failed due to some unspecified error.

Settings

Settings — Sharing settings between applications

Functions

GtkSettings *	gtk_settings_get_default ()
GtkSettings *	gtk_settings_get_for_screen ()
void	gtk_settings_install_property ()
void	gtk_settings_install_property_parser ()
gboolean	gtk_rc_property_parse_color ()
gboolean	gtk_rc_property_parse_enum ()
gboolean	gtk_rc_property_parse_flags ()
gboolean	gtk_rc_property_parse_requisition ()
gboolean	gtk_rc_property_parse_border ()
void	gtk_settings_set_property_value ()
void	gtk_settings_set_string_property ()
void	gtk_settings_set_long_property ()
void	gtk_settings_set_double_property ()
void	gtk_settings_reset_property ()

Properties

GHashTable *	color-hash	Read
gboolean	gtk-alternative-button-order	Read / Write
gboolean	gtk-alternative-sort-arrows	Read / Write
gboolean	gtk-application-prefer-dark-theme	Read / Write
gboolean	gtk-auto-mnemonics	Read / Write
gboolean	gtk-button-images	Read / Write

gboolean	gtk-can-change-accels	Read / Write
gchar *	gtk-color-palette	Read / Write
gchar *	gtk-color-scheme	Read / Write
gboolean	gtk-cursor-blink	Read / Write
gint	gtk-cursor-blink-time	Read / Write
gint	gtk-cursor-blink-timeout	Read / Write
gchar *	gtk-cursor-theme-name	Read / Write
gint	gtk-cursor-theme-size	Read / Write
gchar *	gtk-decoration-layout	Read / Write
gboolean	gtk-dialogs-use-header	Read / Write
gint	gtk-dnd-drag-threshold	Read / Write
gint	gtk-double-click-distance	Read / Write
gint	gtk-double-click-time	Read / Write
gboolean	gtk-enable-accels	Read / Write
gboolean	gtk-enable-animations	Read / Write
gboolean	gtk-enable-event-sounds	Read / Write
gboolean	gtk-enable-input-feedback-sounds	Read / Write
gboolean	gtk-enable-mnemonics	Read / Write
gboolean	gtk-enable-primary-paste	Read / Write
gboolean	gtk-enable-tooltips	Read / Write
guint	gtk-entry-password-hint-timeout	Read / Write
gboolean	gtk-entry-select-on-focus	Read / Write
gboolean	gtk-error-bell	Read / Write
gchar *	gtk-fallback-icon-theme	Read / Write
gchar *	gtk-file-chooser-backend	Read / Write
gchar *	gtk-font-name	Read / Write
guint	gtk-fontconfig-timestamp	Read / Write
gchar *	gtk-icon-sizes	Read / Write
gchar *	gtk-icon-theme-name	Read / Write
gchar *	gtk-im-module	Read / Write
GtkIMPreeditStyle	gtk-im-preedit-style	Read / Write
GtkIMStatusStyle	gtk-im-status-style	Read / Write
gchar *	gtk-key-theme-name	Read / Write
gboolean	gtk-keynav-cursor-only	Read / Write
gboolean	gtk-keynav-use-caret	Read / Write
gboolean	gtk-keynav-wrap-around	Read / Write
gboolean	gtk-label-select-on-focus	Read / Write
guint	gtk-long-press-time	Read / Write
gchar *	gtk-menu-bar-accel	Read / Write
gint	gtk-menu-bar-popup-delay	Read / Write
gboolean	gtk-menu-images	Read / Write
gint	gtk-menu-popdown-delay	Read / Write
gint	gtk-menu-popup-delay	Read / Write
gchar *	gtk-modules	Read / Write
gboolean	gtk-overlay-scrolling	Read / Write
gboolean	gtk-primary-button-warps-slider	Read / Write
gchar *	gtk-print-backends	Read / Write
gchar *	gtk-print-preview-command	Read / Write
gboolean	gtk-recent-files-enabled	Read / Write
gint	gtk-recent-files-limit	Read / Write

gint	gtk-recent-files-max-age	Read / Write
GtkCornerType	gtk-scrolled-window-placement	Read / Write
gboolean	gtk-shell-shows-app-menu	Read / Write
gboolean	gtk-shell-shows-desktop	Read / Write
gboolean	gtk-shell-shows-menubar	Read / Write
gboolean	gtk-show-input-method-menu	Read / Write
gboolean	gtk-show-unicode-menu	Read / Write
gchar *	gtk-sound-theme-name	Read / Write
gboolean	gtk-split-cursor	Read / Write
gchar *	gtk-theme-name	Read / Write
gint	gtk-timeout-expand	Read / Write
gint	gtk-timeout-initial	Read / Write
gint	gtk-timeout-repeat	Read / Write
gchar *	gtk-titlebar-double-click	Read / Write
gchar *	gtk-titlebar-middle-click	Read / Write
gchar *	gtk-titlebar-right-click	Read / Write
GtkIconSize	gtk-toolbar-icon-size	Read / Write
GtkToolbarStyle	gtk-toolbar-style	Read / Write
gint	gtk-tooltip-browse-mode-timeout	Read / Write
gint	gtk-tooltip-browse-timeout	Read / Write
gint	gtk-tooltip-timeout	Read / Write
gboolean	gtk-touchscreen-mode	Read / Write
GtkPolicyType	gtk-visible-focus	Read / Write
gint	gtk-xft-antialias	Read / Write
gint	gtk-xft-dpi	Read / Write
gint	gtk-xft-hinting	Read / Write
gchar *	gtk-xft-hintstyle	Read / Write
gchar *	gtk-xft-rgba	Read / Write

Types and Values

	GtkSettings
struct	GtkSettingsValue
enum	GtkIMPreeditStyle
enum	GtkIMStatusStyle

Object Hierarchy

```
GObject
└── GtkSettings
```

Implemented Interfaces

GtkSettings implements GtkStyleProvider and GtkStyleProviderPrivate.

Includes

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

Description

GtkSettings provide a mechanism to share global settings between applications.

On the X window system, this sharing is realized by an XSettings manager that is usually part of the desktop environment, along with utilities that let the user change these settings. In the absence of an Xsettings manager, GTK+ reads default values for settings from `settings.ini` files in `/etc/gtk-3.0`, `$XDG_CONFIG_DIRS/gtk-3.0` and `$XDG_CONFIG_HOME/gtk-3.0`. These files must be valid key files (see GKeyFile), and have a section called Settings. Themes can also provide default values for settings by installing a `settings.ini` file next to their `gtk.css` file.

Applications can override system-wide settings by setting the property of the GtkSettings object with `g_object_set()`. This should be restricted to special cases though; GtkSettings are not meant as an application configuration facility. When doing so, you need to be aware that settings that are specific to individual widgets may not be available before the widget type has been realized at least once. The following example demonstrates a way to do this:

```
1 gtk_init (&argc, &argv);
2
3 // make sure the type is realized
4 g_type_class_unref (g_type_class_ref (GTK_TYPE_IMAGE_MENU_ITEM));
5
6 g_object_set (gtk_settings_get_default (), "gtk-enable-animations", FALSE, NULL);
```

There is one GtkSettings instance per screen. It can be obtained with `gtk_settings_get_for_screen()`, but in many cases, it is more convenient to use `gtk_widget_get_settings()`. `gtk_settings_get_default()` returns the GtkSettings instance for the default screen.

Functions

gtk_settings_get_default ()

```
GtkSettings *
gtk_settings_get_default (void);
```

Gets the GtkSettings object for the default GDK screen, creating it if necessary. See `gtk_settings_get_for_screen()`.

Returns

a GtkSettings object. If there is no default screen, then returns `NULL`.

[nullable][transfer none]

---

gtk_settings_get_for_screen ()

```
GtkSettings *
gtk_settings_get_for_screen (GdkScreen *screen);
```

Gets the GtkSettings object for `screen` , creating it if necessary.

Parameters

screen	a GdkScreen.

Returns

a GtkSettings object.

[transfer none]

Since: 2.2

---

gtk_settings_install_property ()

```
void
gtk_settings_install_property (GParamSpec *pspec);
```

`gtk_settings_install_property` has been deprecated since version 3.16 and should not be used in newly-written code.

This function is not useful outside GTK+.

---

gtk_settings_install_property_parser ()

```
void
gtk_settings_install_property_parser (GParamSpec *pspec,
                                      GtkRcPropertyParser parser);
```

`gtk_settings_install_property_parser` has been deprecated since version 3.16 and should not be used in newly-written code.

This function is not useful outside GTK+.

Parameters

parser	.	[scope call]

---

gtk_rc_property_parse_color ()

```
gboolean
gtk_rc_property_parse_color (const GParamSpec *pspec,
                             const GString *gstring,
                             GValue *property_value);
```

A GtkRcPropertyParser for use with `gtk_settings_install_property_parser()` or

gtk_widget_class_install_style_property_parser() which parses a color given either by its name or in the form { red, green, blue } where red, green and blue are integers between 0 and 65535 or floating-point numbers between 0 and 1.

Parameters

pspec	a GParamSpec
gstring	the GString to be parsed
property_value	a GValue which must hold GdkColor values.

Returns

TRUE if gstring could be parsed and property_value has been set to the resulting GdkColor.

---

gtk_rc_property_parse_enum ()

```
gboolean
gtk_rc_property_parse_enum (const GParamSpec *pspec,
                            const GString *gstring,
                            GValue *property_value);
```

A GtkRcPropertyParser for use with gtk_settings_install_property_parser() or gtk_widget_class_install_style_property_parser() which parses a single enumeration value.

The enumeration value can be specified by its name, its nickname or its numeric value. For consistency with flags parsing, the value may be surrounded by parentheses.

Parameters

pspec	a GParamSpec
gstring	the GString to be parsed
property_value	a GValue which must hold enum values.

Returns

TRUE if gstring could be parsed and property_value has been set to the resulting GEnumValue.

---

gtk_rc_property_parse_flags ()

```
gboolean
gtk_rc_property_parse_flags (const GParamSpec *pspec,
                             const GString *gstring,
                             GValue *property_value);
```

A GtkRcPropertyParser for use with gtk_settings_install_property_parser() or gtk_widget_class_install_style_property_parser() which parses flags.

Flags can be specified by their name, their nickname or numerically. Multiple flags can be specified in the form `"( flag1 | flag2 | ... )"`.

Parameters

pspec	a GParamSpec
gstring	the GString to be parsed
property_value	a GValue which must hold flags values.

Returns

`TRUE` if `gstring` could be parsed and `property_value` has been set to the resulting flags value.

---

gtk_rc_property_parse_requisition ()

```
gboolean
gtk_rc_property_parse_requisition (const GParamSpec *pspec,
                                   const GString *gstring,
                                   GValue *property_value);
```

A GtkRcPropertyParser for use with `gtk_settings_install_property_parser()` or `gtk_widget_class_install_style_property_parser()` which parses a requisition in the form `"{ width, height }"` for integers `width` and `height`.

Parameters

pspec	a GParamSpec
gstring	the GString to be parsed
property_value	a GValue which must hold boxed values.

Returns

`TRUE` if `gstring` could be parsed and `property_value` has been set to the resulting GtkRequisition.

---

gtk_rc_property_parse_border ()

```
gboolean
gtk_rc_property_parse_border (const GParamSpec *pspec,
                              const GString *gstring,
                              GValue *property_value);
```

A GtkRcPropertyParser for use with `gtk_settings_install_property_parser()` or `gtk_widget_class_install_style_property_parser()` which parses borders in the form `"{ left, right, top, bottom }"` for integers left, right, top and bottom.

Parameters

pspec	a GParamSpec
gstring	the GString to be parsed
property_value	a GValue which must hold boxed values.

Returns

`TRUE` if `gstring` could be parsed and `property_value` has been set to the resulting GtkBorder.

---

gtk_settings_set_property_value ()

```
void
gtk_settings_set_property_value (GtkSettings *settings,
                                 const gchar *name,
                                 const GtkSettingsValue *svalue);
```

`gtk_settings_set_property_value` has been deprecated since version 3.16 and should not be used in newly-written code.

Use `g_object_set()` instead.

---

gtk_settings_set_string_property ()

```
void
gtk_settings_set_string_property (GtkSettings *settings,
                                  const gchar *name,
                                  const gchar *v_string,
                                  const gchar *origin);
```

`gtk_settings_set_string_property` has been deprecated since version 3.16 and should not be used in newly-written code.

Use `g_object_set()` instead.

---

gtk_settings_set_long_property ()

```
void
gtk_settings_set_long_property (GtkSettings *settings,
                                const gchar *name,
                                glong v_long,
                                const gchar *origin);
```

`gtk_settings_set_long_property` has been deprecated since version 3.16 and should not be used in newly-written code.

Use `g_object_set()` instead.

---

gtk_settings_set_double_property ()

```
void
gtk_settings_set_double_property (GtkSettings *settings,
                                  const gchar *name,
                                  gdouble v_double,
                                  const gchar *origin);
```

`gtk_settings_set_double_property` has been deprecated since version 3.16 and should not be used in newly-written code.

Use `g_object_set()` instead.

---

gtk_settings_reset_property ()

```
void
gtk_settings_reset_property (GtkSettings *settings,
                             const gchar *name);
```

Undoes the effect of calling `g_object_set()` to install an application-specific value for a setting. After this call, the setting will again follow the session-wide value for this setting.

Parameters

settings	a GtkSettings object
name	the name of the setting to reset

Since: 3.20

Types and Values

GtkSettings

```
typedef struct _GtkSettings GtkSettings;
```

---

struct GtkSettingsValue

```
struct GtkSettingsValue {
  /* origin should be something like "filename:linenumber" for rc files,
   * or e.g. "XProperty" for other sources
   */
  gchar *origin;

  /* valid types are LONG, DOUBLE and STRING corresponding to the token parsed,
   * or a GSTRING holding an unparsed statement
   */
  GValue value;
};
```

Members

gchar *`origin`;	Origin should be something like “filename:linenumber” for rc files, or e.g. “XProperty” for other sources.
GValue `value`;	Valid types are LONG, DOUBLE and STRING corresponding to the token parsed, or a GSTRING holding an unparsed statement

---

enum GtkIMPreeditStyle

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

Style for input method preedit. See also “gtk-im-preedit-style”

Members

GTK_IM_PREEDIT_NOTHING	Deprecated
GTK_IM_PREEDIT_CALLBACK	Deprecated
GTK_IM_PREEDIT_NONE	Deprecated

---

enum GtkIMStatusStyle

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

Style for input method status. See also “gtk-im-status-style”

Members

GTK_IM_STATUS_NOTHING	Deprecated
GTK_IM_STATUS_CALLBACK	Deprecated
GTK_IM_STATUS_NONE	Deprecated

Property Details

The `“color-hash”` property

```
  “color-hash”               GHashTable *
```

Holds a hash table representation of the “gtk-color-scheme” setting, mapping color names to GdkColors.

[transfer container]

`GtkSettings:color-hash` has been deprecated since version 3.8 and should not be used in newly-written code.

Will always return an empty hash table.

Flags: Read

Since: 2.10

The “gtk-alternative-button-order” property

“gtk-alternative-button-order” gboolean

Whether buttons in dialogs should use the alternative button order.

Flags: Read / Write

Default value: FALSE

The “gtk-alternative-sort-arrows” property

“gtk-alternative-sort-arrows” gboolean

Controls the direction of the sort indicators in sorted list and tree views. By default an arrow pointing down means the column is sorted in ascending order. When set to TRUE, this order will be inverted.

Flags: Read / Write

Default value: FALSE

Since: 2.12

The “gtk-application-prefer-dark-theme” property

“gtk-application-prefer-dark-theme” gboolean

Whether the application prefers to use a dark theme. If a GTK+ theme includes a dark variant, it will be used instead of the configured theme.

Some applications benefit from minimizing the amount of light pollution that interferes with the content. Good candidates for dark themes are photo and video editors that make the actual content get all the attention and minimize the distraction of the chrome.

Dark themes should not be used for documents, where large spaces are white/light and the dark chrome creates too much contrast (web browser, text editor...).

Flags: Read / Write

Default value: FALSE

Since: 3.0

The “gtk-auto-mnemonics” property

“gtk-auto-mnemonics” gboolean

Whether mnemonics should be automatically shown and hidden when the user presses the mnemonic activator.

`GtkSettings:gtk-auto-mnemonics` has been deprecated since version 3.10 and should not be used in newly-written code.

This setting is ignored

Flags: Read / Write

Default value: TRUE

Since: 2.20

The `"gtk-button-images"` property

`"gtk-button-images"` `gboolean`

Whether images should be shown on buttons

`GtkSettings:gtk-button-images` has been deprecated since version 3.10 and should not be used in newly-written code.

This setting is deprecated. Application developers control whether a button should show an icon or not, on a per-button basis. If a GtkButton should show an icon, use the "always-show-image" property of GtkButton, and pack a GtkImage inside the GtkButton

Flags: Read / Write

Default value: FALSE

Since: 2.4

The `"gtk-can-change-accels"` property

`"gtk-can-change-accels"` `gboolean`

Whether menu accelerators can be changed by pressing a key over the menu item.

`GtkSettings:gtk-can-change-accels` has been deprecated since version 3.10 and should not be used in newly-written code.

This setting is ignored.

Flags: Read / Write

Default value: FALSE

The `"gtk-color-palette"` property

`"gtk-color-palette"` `gchar *`

Palette to use in the deprecated color selector.

GtkSettings:gtk-color-palette has been deprecated since version 3.10 and should not be used in newly-written code.

Only used by the deprecated color selector widget.

Flags: Read / Write

Default value: "black:white:gray50:red:purple:blue:light blue:green:yellow:orange:lavender:brown:goldenrod4:dodger blue:pink:light green:gray10:gray30:gray75:gray90"

The “gtk-color-scheme” property

```
  “gtk-color-scheme”         gchar *
```

A palette of named colors for use in themes. The format of the string is

```
1                                       name1: color1
2                                       name2: color2
3                                       ...
```

Color names must be acceptable as identifiers in the gtkrc syntax, and color specifications must be in the format accepted by gdk_color_parse().

Note that due to the way the color tables from different sources are merged, color specifications will be converted to hexadecimal form when getting this property.

Starting with GTK+ 2.12, the entries can alternatively be separated by ';' instead of newlines:

```
1                                       name1: color1; name2: color2; ...
```

GtkSettings:gtk-color-scheme has been deprecated since version 3.8 and should not be used in newly-written code.

Color scheme support was dropped and is no longer supported. You can still set this property, but it will be ignored.

Flags: Read / Write

Default value: ""

Since: 2.10

The “gtk-cursor-blink” property

```
  “gtk-cursor-blink”         gboolean
```

Whether the cursor should blink.

Also see the “gtk-cursor-blink-timeout” setting, which allows more flexible control over cursor blinking.

Flags: Read / Write

Default value: TRUE

The “gtk-cursor-blink-time” property

```
“gtk-cursor-blink-time”    gint
```

Length of the cursor blink cycle, in milliseconds.

Flags: Read / Write

Allowed values: >= 100

Default value: 1200

The “gtk-cursor-blink-timeout” property

```
“gtk-cursor-blink-timeout” gint
```

Time after which the cursor stops blinking, in seconds. The timer is reset after each user interaction.

Setting this to zero has the same effect as setting “gtk-cursor-blink” to FALSE.

Flags: Read / Write

Allowed values: >= 1

Default value: 10

Since: 2.12

The “gtk-cursor-theme-name” property

```
“gtk-cursor-theme-name”    gchar *
```

Name of the cursor theme to use, or NULL to use the default theme.

Flags: Read / Write

Default value: NULL

The “gtk-cursor-theme-size” property

```
“gtk-cursor-theme-size”    gint
```

Size to use for cursors, or 0 to use the default size.

Flags: Read / Write

Allowed values: [0,128]

Default value: 0

The “gtk-decoration-layout” property

“gtk-decoration-layout” gchar *

This setting determines which buttons should be put in the titlebar of client-side decorated windows, and whether they should be placed at the left of right.

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.

Note that buttons will only be shown when they are meaningful. E.g. a menu button only appears when the desktop shell does not show the app menu, and a close button only appears on a window that can be closed.

Also note that the setting can be overridden with the “decoration-layout” property.

Flags: Read / Write

Default value: "menu:minimize,maximize,close"

Since: 3.12

The “gtk-dialogs-use-header” property

“gtk-dialogs-use-header” gboolean

Whether builtin GTK+ dialogs such as the file chooser, the color chooser or the font chooser will use a header bar at the top to show action widgets, or an action area at the bottom.

This setting does not affect custom dialogs using GtkDialog directly, or message dialogs.

Flags: Read / Write

Default value: FALSE

Since: 3.12

The “gtk-dnd-drag-threshold” property

“gtk-dnd-drag-threshold” gint

Number of pixels the cursor can move before dragging.

Flags: Read / Write

Allowed values: >= 1

Default value: 8

The “gtk-double-click-distance” property

“gtk-double-click-distance” gint

Maximum distance allowed between two clicks for them to be considered a double click (in pixels).

Flags: Read / Write

Allowed values: >= 0

Default value: 5

The “gtk-double-click-time” property

“gtk-double-click-time” gint

Maximum time allowed between two clicks for them to be considered a double click (in milliseconds).

Flags: Read / Write

Allowed values: >= 0

Default value: 400

The “gtk-enable-accels” property

“gtk-enable-accels” gboolean

Whether menu items should have visible accelerators which can be activated.

Flags: Read / Write

Default value: TRUE

Since: 2.12

The “gtk-enable-animations” property

“gtk-enable-animations” gboolean

Whether to enable toolkit-wide animations.

Flags: Read / Write

Default value: TRUE

The “gtk-enable-event-sounds” property

“gtk-enable-event-sounds” gboolean

Whether to play any event sounds at all.

See the Sound Theme Specifications for more information on event sounds and sound themes.

GTK+ itself does not support event sounds, you have to use a loadable module like the one that comes with libcanberra.

Flags: Read / Write

Default value: TRUE

Since: 2.14

The “gtk-enable-input-feedback-sounds” property

```
“gtk-enable-input-feedback-sounds” gboolean
```

Whether to play event sounds as feedback to user input.

See the Sound Theme Specifications for more information on event sounds and sound themes.

GTK+ itself does not support event sounds, you have to use a loadable module like the one that comes with libcanberra.

Flags: Read / Write

Default value: TRUE

Since: 2.14

The “gtk-enable-mnemonics” property

```
“gtk-enable-mnemonics”     gboolean
```

Whether labels and menu items should have visible mnemonics which can be activated.

`GtkSettings:gtk-enable-mnemonics` has been deprecated since version 3.10 and should not be used in newly-written code.

This setting can still be used for application overrides, but will be ignored in the future

Flags: Read / Write

Default value: TRUE

Since: 2.12

The “gtk-enable-primary-paste” property

```
“gtk-enable-primary-paste” gboolean
```

Whether a middle click on a mouse should paste the 'PRIMARY' clipboard content at the cursor location.

Flags: Read / Write

Default value: TRUE

Since: 3.4

The “gtk-enable-tooltips” property

“gtk-enable-tooltips” gboolean

Whether tooltips should be shown on widgets.

GtkSettings:gtk-enable-tooltips has been deprecated since version 3.10 and should not be used in newly-written code.

This setting is ignored.

Flags: Read / Write

Default value: TRUE

Since: 2.14

The “gtk-entry-password-hint-timeout” property

“gtk-entry-password-hint-timeout” guint

How long to show the last input character in hidden entries. This value is in milliseconds. 0 disables showing the last char. 600 is a good value for enabling it.

Flags: Read / Write

Default value: 0

Since: 2.10

The “gtk-entry-select-on-focus” property

“gtk-entry-select-on-focus” gboolean

Whether to select the contents of an entry when it is focused.

Flags: Read / Write

Default value: TRUE

The “gtk-error-bell” property

“gtk-error-bell” gboolean

When TRUE, keyboard navigation and other input-related errors will cause a beep. Since the error bell is implemented using gdk_window_beep(), the windowing system may offer ways to configure the error bell in

many ways, such as flashing the window or similar visual effects.

Flags: Read / Write

Default value: TRUE

Since: 2.12

The “gtk-fallback-icon-theme” property

```
“gtk-fallback-icon-theme”  gchar *
```

Name of a icon theme to fall back to.

`GtkSettings:gtk-fallback-icon-theme` has been deprecated since version 3.10 and should not be used in newly-written code.

This setting is ignored.

Flags: Read / Write

Default value: NULL

The “gtk-file-chooser-backend” property

```
“gtk-file-chooser-backend” gchar *
```

Name of the GtkFileChooser backend to use by default.

`GtkSettings:gtk-file-chooser-backend` has been deprecated since version 3.10 and should not be used in newly-written code.

This setting is ignored. GtkFileChooser uses GIO by default.

Flags: Read / Write

Default value: NULL

The “gtk-font-name” property

```
“gtk-font-name”            gchar *
```

The default font to use. GTK+ uses the family name and size from this string.

Flags: Read / Write

Default value: "Sans 10"

The “gtk-fontconfig-timestamp” property

“gtk-fontconfig-timestamp” guint

Timestamp of current fontconfig configuration.

Flags: Read / Write

Default value: 0

The “gtk-icon-sizes” property

“gtk-icon-sizes” gchar *

A list of icon sizes. The list is separated by colons, and item has the form:

size-name = width , height

E.g. "gtk-menu=16,16:gtk-button=20,20:gtk-dialog=48,48". GTK+ itself use the following named icon sizes: gtk-menu, gtk-button, gtk-small-toolbar, gtk-large-toolbar, gtk-dnd, gtk-dialog. Applications can register their own named icon sizes with gtk_icon_size_register().

GtkSettings:gtk-icon-sizes has been deprecated since version 3.10 and should not be used in newly-written code.

This setting is ignored.

Flags: Read / Write

Default value: NULL

The “gtk-icon-theme-name” property

“gtk-icon-theme-name” gchar *

Name of icon theme to use.

Flags: Read / Write

Default value: "Adwaita"

The “gtk-im-module” property

“gtk-im-module” gchar *

Which IM (input method) module should be used by default. This is the input method that will be used if the user has not explicitly chosen another input method from the IM context menu. This also can be a colon-separated list of input methods, which GTK+ will try in turn until it finds one available on the system.

See GtkIMContext.

Flags: Read / Write

Default value: NULL

The “gtk-im-preedit-style” property

“gtk-im-preedit-style” GtkIMPreeditStyle

How to draw the input method preedit string.

GtkSettings:gtk-im-preedit-style has been deprecated since version 3.10 and should not be used in newly-written code.

This setting is ignored.

Flags: Read / Write

Default value: GTK_IM_PREEDIT_CALLBACK

The “gtk-im-status-style” property

“gtk-im-status-style” GtkIMStatusStyle

How to draw the input method statusbar.

GtkSettings:gtk-im-status-style has been deprecated since version 3.10 and should not be used in newly-written code.

This setting is ignored.

Flags: Read / Write

Default value: GTK_IM_STATUS_CALLBACK

The “gtk-key-theme-name” property

“gtk-key-theme-name” gchar *

Name of key theme to load.

Flags: Read / Write

Default value: NULL

The “gtk-keynav-cursor-only” property

“gtk-keynav-cursor-only” gboolean

When TRUE, keyboard navigation should be able to reach all widgets by using the cursor keys only. Tab, Shift etc. keys can't be expected to be present on the used input device.

GtkSettings:gtk-keynav-cursor-only has been deprecated since version 3.10 and should not be used in newly-written code.

Generally, the behavior for touchscreen input should be performed dynamically based on `gdk_event_get_source_device()`.

Flags: Read / Write

Default value: FALSE

Since: 2.12

The “gtk-keynav-use-caret” property

`“gtk-keynav-use-caret” gboolean`

Whether GTK+ should make sure that text can be navigated with a caret, even if it is not editable. This is useful when using a screen reader.

Flags: Read / Write

Default value: FALSE

Since: 3.20

The “gtk-keynav-wrap-around” property

`“gtk-keynav-wrap-around” gboolean`

When `TRUE`, some widgets will wrap around when doing keyboard navigation, such as menus, menubars and notebooks.

`GtkSettings:gtk-keynav-wrap-around` has been deprecated since version 3.10 and should not be used in newly-written code.

This setting is ignored.

Flags: Read / Write

Default value: TRUE

Since: 2.12

The “gtk-label-select-on-focus” property

`“gtk-label-select-on-focus” gboolean`

Whether to select the contents of a selectable label when it is focused.

Flags: Read / Write

Default value: TRUE

The “gtk-long-press-time” property

“gtk-long-press-time” guint

The time for a button or touch press to be considered a "long press".

Flags: Read / Write

Allowed values: <= G_MAXINT

Default value: 500

Since: 3.14

The “gtk-menu-bar-accel” property

“gtk-menu-bar-accel” gchar *

Keybinding to activate the menu bar.

GtkSettings:gtk-menu-bar-accel has been deprecated since version 3.10 and should not be used in newly-written code.

This setting can still be used for application overrides, but will be ignored in the future

Flags: Read / Write

Default value: "F10"

The “gtk-menu-bar-popup-delay” property

“gtk-menu-bar-popup-delay” gint

Delay before the submenus of a menu bar appear.

GtkSettings:gtk-menu-bar-popup-delay has been deprecated since version 3.10 and should not be used in newly-written code.

This setting is ignored.

Flags: Read / Write

Allowed values: >= 0

Default value: 0

The “gtk-menu-images” property

“gtk-menu-images” gboolean

Whether images should be shown in menu items

GtkSettings:gtk-menu-images has been deprecated since version 3.10 and should not be used in newly-written code.

This setting is deprecated. Application developers control whether or not a GtkMenuItem should have an icon or not, on a per widget basis. Either use a GtkMenuItem with a GtkBox containing a GtkImage and a GtkAccelLabel, or describe your menus using a GMenu XML description

Flags: Read / Write

Default value: FALSE

The “gtk-menu-popdown-delay” property

“gtk-menu-popdown-delay” gint

The time before hiding a submenu when the pointer is moving towards the submenu.

GtkSettings:gtk-menu-popdown-delay has been deprecated since version 3.10 and should not be used in newly-written code.

This setting is ignored.

Flags: Read / Write

Allowed values: >= 0

Default value: 1000

The “gtk-menu-popup-delay” property

“gtk-menu-popup-delay” gint

Minimum time the pointer must stay over a menu item before the submenu appear.

GtkSettings:gtk-menu-popup-delay has been deprecated since version 3.10 and should not be used in newly-written code.

This setting is ignored.

Flags: Read / Write

Allowed values: >= 0

Default value: 225

The “gtk-modules” property

“gtk-modules” gchar *

List of currently active GTK modules.

Flags: Read / Write

Default value: NULL

The “gtk-overlay-scrolling” property

“gtk-overlay-scrolling” gboolean

Whether scrolled windows may use overlayed scrolling indicators. If this is set to FALSE, scrolled windows will have permanent scrollbars.

Flags: Read / Write

Default value: TRUE

Since: 3.24.9

The “gtk-primary-button-warps-slider” property

“gtk-primary-button-warps-slider” gboolean

If the value of this setting is TRUE, clicking the primary button in a GtkRange trough will move the slider, and hence set the range’s value, to the point that you clicked. If it is FALSE, a primary click will cause the slider/value to move by the range’s page-size towards the point clicked.

Whichever action you choose for the primary button, the other action will be available by holding Shift and primary-clicking, or (since GTK+ 3.22.25) clicking the middle mouse button.

Flags: Read / Write

Default value: TRUE

Since: 3.6

The “gtk-print-backends” property

“gtk-print-backends” gchar *

A comma-separated list of print backends to use in the print dialog. Available print backends depend on the GTK+ installation, and may include "file", "cups", "lpr" or "papi".

Flags: Read / Write

Default value: "file,cups"

Since: 2.10

The “gtk-print-preview-command” property

“gtk-print-preview-command” gchar *

A command to run for displaying the print preview. The command should contain a %f placeholder, which will get replaced by the path to the pdf file. The command may also contain a %s placeholder, which will get replaced by the path to a file containing the print settings in the format produced by

gtk_print_settings_to_file().

The preview application is responsible for removing the pdf file and the print settings file when it is done.

Flags: Read / Write

Default value: "evince --unlink-tempfile --preview --print-settings %s %f"

Since: 2.10

The “gtk-recent-files-enabled” property

“gtk-recent-files-enabled” gboolean

Whether GTK+ should keep track of items inside the recently used resources list. If set to FALSE, the list will always be empty.

Flags: Read / Write

Default value: TRUE

Since: 3.8

The “gtk-recent-files-limit” property

“gtk-recent-files-limit” gint

The number of recently used files that should be displayed by default by GtkRecentChooser implementations and by the GtkFileChooser. A value of -1 means every recently used file stored.

GtkSettings:gtk-recent-files-limit has been deprecated since version 3.10 and should not be used in newly-written code.

This setting is ignored

Flags: Read / Write

Allowed values: >= -1

Default value: 50

Since: 2.12

The “gtk-recent-files-max-age” property

“gtk-recent-files-max-age” gint

The maximum age, in days, of the items inside the recently used resources list. Items older than this setting will be excised from the list. If set to 0, the list will always be empty; if set to -1, no item will be removed.

Flags: Read / Write

Allowed values: >= -1

Default value: 30

Since: 2.14

The “gtk-scrolled-window-placement” property

“gtk-scrolled-window-placement” GtkCornerType

Where the contents of scrolled windows are located with respect to the scrollbars, if not overridden by the scrolled window's own placement.

GtkSettings:gtk-scrolled-window-placement has been deprecated since version 3.10 and should not be used in newly-written code.

This setting is ignored.

Flags: Read / Write

Default value: GTK_CORNER_TOP_LEFT

Since: 2.10

The “gtk-shell-shows-app-menu” property

“gtk-shell-shows-app-menu” gboolean

Set to TRUE if the desktop environment is displaying the app menu, FALSE if the app should display it itself.

Flags: Read / Write

Default value: FALSE

The “gtk-shell-shows-desktop” property

“gtk-shell-shows-desktop” gboolean

Set to TRUE if the desktop environment is displaying the desktop folder, FALSE if not.

Flags: Read / Write

Default value: TRUE

The “gtk-shell-shows-menubar” property

“gtk-shell-shows-menubar” gboolean

Set to TRUE if the desktop environment is displaying the menubar, FALSE if the app should display it itself.

Flags: Read / Write

Default value: FALSE

The “gtk-show-input-method-menu” property

“gtk-show-input-method-menu” gboolean

Whether the context menus of entries and text views should offer to change the input method.

GtkSettings:gtk-show-input-method-menu has been deprecated since version 3.10 and should not be used in newly-written code.

This setting is ignored.

Flags: Read / Write

Default value: FALSE

The “gtk-show-unicode-menu” property

“gtk-show-unicode-menu” gboolean

Whether the context menus of entries and text views should offer to insert control characters.

GtkSettings:gtk-show-unicode-menu has been deprecated since version 3.10 and should not be used in newly-written code.

This setting is ignored.

Flags: Read / Write

Default value: FALSE

The “gtk-sound-theme-name” property

“gtk-sound-theme-name” gchar *

The XDG sound theme to use for event sounds.

See the Sound Theme Specifications for more information on event sounds and sound themes.

GTK+ itself does not support event sounds, you have to use a loadable module like the one that comes with libcanberra.

Flags: Read / Write

Default value: "freedesktop"

Since: 2.14

The “gtk-split-cursor” property

“gtk-split-cursor” gboolean

Whether two cursors should be displayed for mixed left-to-right and right-to-left text.

Flags: Read / Write

Default value: TRUE

The “gtk-theme-name” property

“gtk-theme-name” gchar *

Name of theme to load.

Flags: Read / Write

Default value: "Adwaita"

The “gtk-timeout-expand” property

“gtk-timeout-expand” gint

Expand value for timeouts, when a widget is expanding a new region.

GtkSettings:gtk-timeout-expand has been deprecated since version 3.10 and should not be used in newly-written code.

This setting is ignored.

Flags: Read / Write

Allowed values: >= 0

Default value: 500

The “gtk-timeout-initial” property

“gtk-timeout-initial” gint

Starting value for timeouts, when button is pressed.

GtkSettings:gtk-timeout-initial has been deprecated since version 3.10 and should not be used in newly-written code.

This setting is ignored.

Flags: Read / Write

Allowed values: >= 0

Default value: 500

The “gtk-timeout-repeat” property

```
“gtk-timeout-repeat”       gint
```

Repeat value for timeouts, when button is pressed.

`GtkSettings:gtk-timeout-repeat` has been deprecated since version 3.10 and should not be used in newly-written code.

This setting is ignored.

Flags: Read / Write

Allowed values: >= 0

Default value: 50

The “gtk-titlebar-double-click” property

```
“gtk-titlebar-double-click” gchar *
```

This setting determines the action to take when a double-click occurs on the titlebar of client-side decorated windows.

Recognized actions are minimize, toggle-maximize, menu, lower or none.

Flags: Read / Write

Default value: "toggle-maximize"

Since: 3.14

The “gtk-titlebar-middle-click” property

```
“gtk-titlebar-middle-click” gchar *
```

This setting determines the action to take when a middle-click occurs on the titlebar of client-side decorated windows.

Recognized actions are minimize, toggle-maximize, menu, lower or none.

Flags: Read / Write

Default value: "none"

Since: 3.14

The “gtk-titlebar-right-click” property

```
“gtk-titlebar-right-click” gchar *
```

This setting determines the action to take when a right-click occurs on the titlebar of client-side decorated windows.

Recognized actions are minimize, toggle-maximize, menu, lower or none.

Flags: Read / Write

Default value: "menu"

Since: 3.14

The “gtk-toolbar-icon-size” property

“gtk-toolbar-icon-size” GtkIconSize

The size of icons in default toolbars.

GtkSettings:gtk-toolbar-icon-size has been deprecated since version 3.10 and should not be used in newly-written code.

This setting is ignored.

Flags: Read / Write

Default value: GTK_ICON_SIZE_LARGE_TOOLBAR

The “gtk-toolbar-style” property

“gtk-toolbar-style” GtkToolbarStyle

The size of icons in default toolbars.

GtkSettings:gtk-toolbar-style has been deprecated since version 3.10 and should not be used in newly-written code.

This setting is ignored.

Flags: Read / Write

Default value: GTK_TOOLBAR_BOTH_HORIZ

The “gtk-tooltip-browse-mode-timeout” property

“gtk-tooltip-browse-mode-timeout” gint

Amount of time, in milliseconds, after which the browse mode will be disabled.

See “gtk-tooltip-browse-timeout” for more information about browse mode.

GtkSettings:gtk-tooltip-browse-mode-timeout has been deprecated since version 3.10 and should not be used in newly-written code.

This setting is ignored.

Flags: Read / Write

Allowed values: >= 0

Default value: 500

Since: 2.12

The “gtk-tooltip-browse-timeout” property

“gtk-tooltip-browse-timeout” gint

Controls the time after which tooltips will appear when browse mode is enabled, in milliseconds.

Browse mode is enabled when the mouse pointer moves off an object where a tooltip was currently being displayed. If the mouse pointer hits another object before the browse mode timeout expires (see “gtk-tooltip-browse-mode-timeout”), it will take the amount of milliseconds specified by this setting to popup the tooltip for the new object.

GtkSettings:gtk-tooltip-browse-timeout has been deprecated since version 3.10 and should not be used in newly-written code.

This setting is ignored.

Flags: Read / Write

Allowed values: >= 0

Default value: 60

Since: 2.12

The “gtk-tooltip-timeout” property

“gtk-tooltip-timeout” gint

Time, in milliseconds, after which a tooltip could appear if the cursor is hovering on top of a widget.

GtkSettings:gtk-tooltip-timeout has been deprecated since version 3.10 and should not be used in newly-written code.

This setting is ignored.

Flags: Read / Write

Allowed values: >= 0

Default value: 500

Since: 2.12

The “gtk-touchscreen-mode” property

“gtk-touchscreen-mode” gboolean

When TRUE, there are no motion notify events delivered on this screen, and widgets can't use the pointer hovering them for any essential functionality.

GtkSettings:gtk-touchscreen-mode has been deprecated since version 3.4. and should not be used in newly-written code.

Generally, the behavior for touchscreen input should be performed dynamically based on gdk_event_get_source_device().

Flags: Read / Write

Default value: FALSE

Since: 2.10

The “gtk-visible-focus” property

“gtk-visible-focus” GtkPolicyType

Whether 'focus rectangles' should be always visible, never visible, or hidden until the user starts to use the keyboard.

GtkSettings:gtk-visible-focus has been deprecated since version 3.10 and should not be used in newly-written code.

This setting is ignored

Flags: Read / Write

Default value: GTK_POLICY_AUTOMATIC

Since: 3.2

The “gtk-xft-antialias” property

“gtk-xft-antialias” gint

Whether to antialias Xft fonts; 0=no, 1=yes, -1=default.

Flags: Read / Write

Allowed values: [-1,1]

Default value: -1

The “gtk-xft-dpi” property

“gtk-xft-dpi” gint

Resolution for Xft, in 1024 * dots/inch. -1 to use default value.

Flags: Read / Write

Allowed values: [-1,1048576]

Default value: -1

The “gtk-xft-hinting” property

“gtk-xft-hinting” gint

Whether to hint Xft fonts; 0=no, 1=yes, -1=default.

Flags: Read / Write

Allowed values: [-1,1]

Default value: -1

The “gtk-xft-hintstyle” property

“gtk-xft-hintstyle” gchar *

What degree of hinting to use; hintnone, hintslight, hintmedium, or hintfull.

Flags: Read / Write

Default value: NULL

The “gtk-xft-rgba” property

“gtk-xft-rgba” gchar *

Type of subpixel antialiasing; none, rgb, bgr, vrgb, vbgr.

Flags: Read / Write

Default value: NULL

Bindings

Bindings — Key bindings for individual widgets

Functions

void	gtk_binding_entry_add_signall ()

GtkBindingSet *	gtk_binding_set_new ()
GtkBindingSet *	gtk_binding_set_by_class ()
GtkBindingSet *	gtk_binding_set_find ()
gboolean	gtk_bindings_activate ()
gboolean	gtk_bindings_activate_event ()
gboolean	gtk_binding_set_activate ()
void	gtk_binding_entry_add_signal ()
GTokenType	gtk_binding_entry_add_signal_from_string ()
void	gtk_binding_entry_skip ()
void	gtk_binding_entry_remove ()
void	gtk_binding_set_add_path ()

Types and Values

struct	GtkBindingSet
struct	GtkBindingEntry
struct	GtkBindingSignal
struct	GtkBindingArg

Includes

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

Description

GtkBindingSet provides a mechanism for configuring GTK+ key bindings through CSS files. This eases key binding adjustments for application developers as well as users and provides GTK+ users or administrators with high key binding configurability which requires no application or toolkit side changes.

In order for bindings to work in a custom widget implementation, the widget's "can-focus" and "has-focus" properties must both be true. For example, by calling `gtk_widget_set_can_focus()` in the widget's initialisation function; and by calling `gtk_widget_grab_focus()` when the widget is clicked.

Installing a key binding

A CSS file binding consists of a "binding-set" definition and a match statement to apply the binding set to specific widget types. Details on the matching mechanism are described under Selectors in the GtkCssProvider documentation. Inside the binding set definition, key combinations are bound to one or more specific signal emissions on the target widget. Key combinations are strings consisting of an optional GdkModifierType name and key names such as those defined in `gdk/gdkkeysyms.h` or returned from `gdk_keyval_name()`, they have to be parsable by `gtk_accelerator_parse()`. Specifications of signal emissions consist of a string identifying the signal name, and a list of signal specific arguments in parenthesis.

For example for binding Control and the left or right cursor keys of a GtkEntry widget to the "move-cursor" signal (so movement occurs in 3-character steps), the following binding can be used:

```
1  @binding-set MoveCursor3
2  {
3    bind "<Control>Right" { "move-cursor" (visual-positions, 3, 0) };
4    bind "<Control>Left" { "move-cursor" (visual-positions, -3, 0) };
```

```
5  }
6
7  entry
8  {
9    -gtk-key-bindings: MoveCursor3;
10 }
```

Unbinding existing key bindings

GTK+ already defines a number of useful bindings for the widgets it provides. Because custom bindings set up in CSS files take precedence over the default bindings shipped with GTK+, overriding existing bindings as demonstrated in Installing a key binding works as expected. The same mechanism can not be used to “unbind” existing bindings, however.

```
1     @binding-set MoveCursor3
2     {
3       bind "<Control>Right" {  };
4       bind "<Control>Left" {  };
5     }
6
7     entry
8     {
9       -gtk-key-bindings: MoveCursor3;
10    }
```

The above example will not have the desired effect of causing “<Control>Right” and “<Control>Left” key presses to be ignored by GTK+. Instead, it just causes any existing bindings from the bindings set “MoveCursor3” to be deleted, so when “<Control>Right” or “<Control>Left” are pressed, no binding for these keys is found in binding set “MoveCursor3”. GTK+ will thus continue to search for matching key bindings, and will eventually lookup and find the default GTK+ bindings for entries which implement word movement. To keep GTK+ from activating its default bindings, the “unbind” keyword can be used like this:

```
1                                                 @binding-set MoveCursor3
2                                                 {
3                                                   unbind "<Control>Right";
4                                                   unbind "<Control>Left";
5                                                 }
6
7                                                 entry
8                                                 {
9                                                   -gtk-key-bindings: MoveCursor3;
10                                                }
```

Now, GTK+ will find a match when looking up “<Control>Right” and “<Control>Left” key presses before it resorts to its default bindings, and the match instructs it to abort (“unbind”) the search, so the key presses are not consumed by this widget. As usual, further processing of the key presses, e.g. by an entry’s parent widget, is now possible.

Functions

gtk_binding_entry_add_signall ()

```
void
gtk_binding_entry_add_signall (GtkBindingSet *binding_set,
```

```
                                guint keyval,
                                GdkModifierType modifiers,
                                const gchar *signal_name,
                                GSList *binding_args);
```

Override or install a new key binding for `keyval` with `modifiers` on `binding_set` .

Parameters

binding_set	a GtkBindingSet to add a signal to	
keyval	key value	
modifiers	key modifier	
signal_name	signal name to be bound	
binding_args	list of GtkBindingArg signal arguments.	[transfer none][element-type GtkBindingArg]

---

gtk_binding_set_new ()

```
GtkBindingSet *
gtk_binding_set_new (const gchar *set_name);
```

GTK+ maintains a global list of binding sets. Each binding set has a unique name which needs to be specified upon creation.

[skip]

Parameters

set_name	unique name of this binding set

Returns

new binding set.

[transfer none]

---

gtk_binding_set_by_class ()

```
GtkBindingSet *
gtk_binding_set_by_class (gpointer object_class);
```

This function returns the binding set named after the type name of the passed in class structure. New binding sets are created on demand by this function.

[skip]

Parameters

object_class	a valid GObject class

Returns

the binding set corresponding to `object_class`.

[transfer none]

---

gtk_binding_set_find ()

```
GtkBindingSet *
gtk_binding_set_find (const gchar *set_name);
```

Find a binding set by its globally unique name.

The `set_name` can either be a name used for `gtk_binding_set_new()` or the type name of a class used in `gtk_binding_set_by_class()`.

Parameters

set_name	unique binding set name

Returns

`NULL` or the specified binding set.

[nullable][transfer none]

---

gtk_bindings_activate ()

```
gboolean
gtk_bindings_activate (GObject *object,
                       guint keyval,
                       GdkModifierType modifiers);
```

Find a key binding matching `keyval` and `modifiers` and activate the binding on `object`.

Parameters

object	object to activate when binding found
keyval	key value of the binding
modifiers	key modifier of the binding

Returns

`TRUE` if a binding was found and activated

---

gtk_bindings_activate_event ()

```
gboolean
gtk_bindings_activate_event (GObject *object,
                             GdkEventKey *event);
```

Looks up key bindings for `object` to find one matching `event` , and if one was found, activate it.

Parameters

object	a GObject (generally must be a widget)
event	a GdkEventKey

Returns

`TRUE` if a matching key binding was found

Since: 2.4

---

gtk_binding_set_activate ()

```
gboolean
gtk_binding_set_activate (GtkBindingSet *binding_set,
                          guint keyval,
                          GdkModifierType modifiers,
                          GObject *object);
```

Find a key binding matching `keyval` and `modifiers` within `binding_set` and activate the binding on `object` .

Parameters

binding_set	a GtkBindingSet set to activate
keyval	key value of the binding
modifiers	key modifier of the binding
object	object to activate when binding found

Returns

`TRUE` if a binding was found and activated

---

gtk_binding_entry_add_signal ()

```
void
gtk_binding_entry_add_signal (GtkBindingSet *binding_set,
                              guint keyval,
```

```
                                GdkModifierType modifiers,
                                const gchar *signal_name,
                                guint n_args,
                                ...);
```

Override or install a new key binding for `keyval` with `modifiers` on `binding_set` . When the binding is activated, `signal_name` will be emitted on the target widget, with `n_args` `Varargs` used as arguments.

Each argument to the signal must be passed as a pair of varargs: the GType of the argument, followed by the argument value (which must be of the given type). There must be `n_args` pairs in total.

Adding a Key Binding

```
1  GtkBindingSet *binding_set;
2  GdkModifierType modmask = GDK_CONTROL_MASK;
3  int count = 1;
4  gtk_binding_entry_add_signal (binding_set,
5                                GDK_KEY_space,
6                                modmask,
7                                "move-cursor", 2,
8                                GTK_TYPE_MOVEMENT_STEP, GTK_MOVEMENT_PAGES,
9                                G_TYPE_INT, count,
10                               G_TYPE_BOOLEAN, FALSE);
```

Parameters

binding_set	a GtkBindingSet to install an entry for
keyval	key value of binding to install
modifiers	key modifier of binding to install
signal_name	signal to execute upon activation
n_args	number of arguments to `signal_name`
...	arguments to `signal_name`

---

gtk_binding_entry_add_signal_from_string ()

```
GTokenType
gtk_binding_entry_add_signal_from_string
                                (GtkBindingSet *binding_set,
                                 const gchar *signal_desc);
```

Parses a signal description from `signal_desc` and incorporates it into `binding_set` .

Signal descriptions may either bind a key combination to one or more signals:

```
1                                       bind "key" {
2                                         "signalname" (param, ...)
3                                         ...
4                                       }
```

Or they may also unbind a key combination:

```
1                                       unbind "key"
```

Key combinations must be in a format that can be parsed by gtk_accelerator_parse().

Parameters

binding_set	a GtkBindingSet
signal_desc	a signal description

Returns

`G_TOKEN_NONE` if the signal was successfully parsed and added, the expected token otherwise

Since: 3.0

---

gtk_binding_entry_skip ()

```
void
gtk_binding_entry_skip (GtkBindingSet *binding_set,
                        guint keyval,
                        GdkModifierType modifiers);
```

Install a binding on `binding_set` which causes key lookups to be aborted, to prevent bindings from lower priority sets to be activated.

Parameters

binding_set	a GtkBindingSet to skip an entry of
keyval	key value of binding to skip
modifiers	key modifier of binding to skip

Since: 2.12

---

gtk_binding_entry_remove ()

```
void
gtk_binding_entry_remove (GtkBindingSet *binding_set,
                          guint keyval,
                          GdkModifierType modifiers);
```

Remove a binding previously installed via `gtk_binding_entry_add_signal()` on `binding_set` .

Parameters

binding_set	a GtkBindingSet to remove an entry of
keyval	key value of binding to remove
modifiers	key modifier of binding to remove

---

gtk_binding_set_add_path ()

```
void
```

```
gtk_binding_set_add_path (GtkBindingSet *binding_set,
                          GtkPathType path_type,
                          const gchar *path_pattern,
                          GtkPathPriorityType priority);
```

`gtk_binding_set_add_path` has been deprecated since version 3.0 and should not be used in newly-written code.

This function was used internally by the GtkRC parsing mechanism to assign match patterns to GtkBindingSet structures.

In GTK+ 3, these match patterns are unused.

Parameters

binding_set	a GtkBindingSet to add a path to
path_type	path type the pattern applies to
path_pattern	the actual match pattern
priority	binding priority

Types and Values

struct GtkBindingSet

```
struct GtkBindingSet {
  gchar           *set_name;
  gint             priority;
  GSList          *widget_path_pspecs;
  GSList          *widget_class_pspecs;
  GSList          *class_branch_pspecs;
  GtkBindingEntry *entries;
  GtkBindingEntry *current;
  guint            parsed : 1;
};
```

A binding set maintains a list of activatable key bindings. A single binding set can match multiple types of widgets. Similar to style contexts, can be matched by any information contained in a widgets GtkWidgetPath. When a binding within a set is matched upon activation, an action signal is emitted on the target widget to carry out the actual activation.

Members

gchar *`set_name`;	unique name of this binding set
gint `priority`;	unused
GSList *`widget_path_pspecs`;	unused
GSList *`widget_class_pspecs`;	unused
GSList *`class_branch_pspecs`;	unused
GtkBindingEntry *`entries`;	the key binding entries in this binding set
GtkBindingEntry *`current`;	implementation detail
guint `parsed` : 1;	whether this binding set stems from a CSS file and is reset upon theme changes

struct GtkBindingEntry

```
struct GtkBindingEntry {
  /* key portion */
  guint             keyval;
  GdkModifierType   modifiers;

  GtkBindingSet    *binding_set;
  guint             destroyed     : 1;
  guint             in_emission   : 1;
  guint             marks_unbound : 1;
  GtkBindingEntry  *set_next;
  GtkBindingEntry  *hash_next;
  GtkBindingSignal *signals;
};
```

Each key binding element of a binding sets binding list is represented by a GtkBindingEntry.

Members

guint `keyval`;	key value to match
GdkModifierType `modifiers`;	key modifiers to match
GtkBindingSet *`binding_set`;	binding set this entry belongs to
guint `destroyed` : 1;	implementation detail
guint `in_emission` : 1;	implementation detail
guint `marks_unbound` : 1;	implementation detail
GtkBindingEntry *`set_next`;	linked list of entries maintained by binding set
GtkBindingEntry *`hash_next`;	implementation detail
GtkBindingSignal *`signals`;	action signals of this entry

struct GtkBindingSignal

```
struct GtkBindingSignal {
  GtkBindingSignal *next;
  gchar            *signal_name;
  guint             n_args;
  GtkBindingArg    *args;
};
```

A GtkBindingSignal stores the necessary information to activate a widget in response to a key press via a signal emission.

Members

GtkBindingSignal *`next`;	implementation detail	
gchar *`signal_name`;	the action signal to be emitted	
guint `n_args`;	number of arguments specified for the signal	
GtkBindingArg *`args`;	the arguments specified for the signal.	[array length=n_args]

struct GtkBindingArg

```
struct GtkBindingArg {
  GType      arg_type;
  union {
    glong    long_data;
    gdouble  double_data;
    gchar   *string_data;
  } d;
};
```

A GtkBindingArg holds the data associated with an argument for a key binding signal emission as stored in GtkBindingSignal.

Members

GType `arg_type`;	implementation detail

See Also

Keyboard Accelerators, Mnemonics, GtkCssProvider

Standard Enumerations

Standard Enumerations — Public enumerated types used throughout GTK+

Types and Values

enum	GtkBaselinePosition
enum	GtkDeleteType
enum	GtkDirectionType
enum	GtkJustification
enum	GtkMovementStep
enum	GtkOrientation
enum	GtkPackType
enum	GtkPositionType
enum	GtkReliefStyle
enum	GtkScrollStep
enum	GtkScrollType
enum	GtkSelectionMode
enum	GtkShadowType
enum	GtkStateFlags
enum	GtkToolbarStyle
enum	GtkSortType

Includes

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

Description

Functions

Types and Values

enum GtkBaselinePosition

Whenever a container has some form of natural row it may align children in that row along a common typographical baseline. If the amount of verical space in the row is taller than the total requested height of the baseline-aligned children then it can use a GtkBaselinePosition to select where to put the baseline inside the extra availible space.

Members

GTK_BASELINE_POSITION_TOP	Align the baseline at the top
GTK_BASELINE_POSITION_CENTER	Center the baseline
GTK_BASELINE_POSITION_BOTTOM	Align the baseline at the bottom

Since: 3.10

---

enum GtkDeleteType

See also: “delete-from-cursor”.

Members

GTK_DELETE_CHARS	Delete characters.
GTK_DELETE_WORD_ENDS	Delete only the portion of the word to the left/right of cursor if we’re in the middle of a word.
GTK_DELETE_WORDS	Delete words.
GTK_DELETE_DISPLAY_LINES	Delete display-lines. Display-lines refers to the visible lines, with respect to to the current line breaks. As opposed to paragraphs, which are defined by line breaks in the input.
GTK_DELETE_DISPLAY_LINE_ENDS	Delete only the portion of the display-line to the left/right of cursor.
GTK_DELETE_PARAGRAPH_ENDS	Delete to the end of the paragraph. Like C-k in Emacs (or its

	reverse).
GTK_DELETE_PARAGRAPHS	Delete entire line. Like C-k in pico.
GTK_DELETE_WHITESPACE	Delete only whitespace. Like M-\ in Emacs.

enum GtkDirectionType

Focus movement types.

Members

GTK_DIR_TAB_FORWARD	Move forward.
GTK_DIR_TAB_BACKWARD	Move backward.
GTK_DIR_UP	Move up.
GTK_DIR_DOWN	Move down.
GTK_DIR_LEFT	Move left.
GTK_DIR_RIGHT	Move right.

enum GtkJustification

Used for justifying the text inside a GtkLabel widget. (See also GtkAlignment).

Members

GTK_JUSTIFY_LEFT	The text is placed at the left edge of the label.
GTK_JUSTIFY_RIGHT	The text is placed at the right edge of the label.
GTK_JUSTIFY_CENTER	The text is placed in the center of the label.
GTK_JUSTIFY_FILL	The text is placed is distributed across the label.

enum GtkMovementStep

Members

GTK_MOVEMENT_LOGICAL_POSITIONS	Move forward or back by graphemes
GTK_MOVEMENT_VISUAL_POSITIONS	Move left or right by graphemes
GTK_MOVEMENT_WORDS	Move forward or back by words
GTK_MOVEMENT_DISPLAY_LINES	Move up or down lines (wrapped lines)
GTK_MOVEMENT_DISPLAY_LINE_ENDS	Move to either end of a line
GTK_MOVEMENT_PARAGRAPHS	Move up or down paragraphs (newline-ended lines)
GTK_MOVEMENT_PARAGRAPH_ENDS	Move to either end of a paragraph
GTK_MOVEMENT_PAGES	Move by pages

GTK_MOVEMENT_BUFFER_ENDS	Move to ends of the buffer
GTK_MOVEMENT_HORIZONTAL_PAGES	Move horizontally by pages

enum GtkOrientation

Represents the orientation of widgets and other objects which can be switched between horizontal and vertical orientation on the fly, like GtkToolbar or GtkGesturePan.

Members

GTK_ORIENTATION_HORIZONTAL	The element is in horizontal orientation.
GTK_ORIENTATION_VERTICAL	The element is in vertical orientation.

enum GtkPackType

Represents the packing location GtkBox children. (See: GtkVBox, GtkHBox, and GtkButtonBox).

Members

GTK_PACK_START	The child is packed into the start of the box
GTK_PACK_END	The child is packed into the end of the box

enum GtkPositionType

Describes which edge of a widget a certain feature is positioned at, e.g. the tabs of a GtkNotebook, the handle of a GtkHandleBox or the label of a GtkScale.

Members

GTK_POS_LEFT	The feature is at the left edge.
GTK_POS_RIGHT	The feature is at the right edge.
GTK_POS_TOP	The feature is at the top edge.
GTK_POS_BOTTOM	The feature is at the bottom edge.

enum GtkReliefStyle

Indicated the relief to be drawn around a GtkButton.

Members

GTK_RELIEF_NORMAL	Draw a normal relief.
GTK_RELIEF_HALF	A half relief. Deprecated in 3.14, does the same as `GTK_RELIEF_NORMAL`
GTK_RELIEF_NONE	No relief.

enum GtkScrollStep

Members

GTK_SCROLL_STEPS	Scroll in steps.
GTK_SCROLL_PAGES	Scroll by pages.
GTK_SCROLL_ENDS	Scroll to ends.
GTK_SCROLL_HORIZONTAL_STEPS	Scroll in horizontal steps.
GTK_SCROLL_HORIZONTAL_PAGES	Scroll by horizontal pages.
GTK_SCROLL_HORIZONTAL_ENDS	Scroll to the horizontal ends.

enum GtkScrollType

Scrolling types.

Members

GTK_SCROLL_NONE	No scrolling.
GTK_SCROLL_JUMP	Jump to new location.
GTK_SCROLL_STEP_BACKWARD	Step backward.
GTK_SCROLL_STEP_FORWARD	Step forward.
GTK_SCROLL_PAGE_BACKWARD	Page backward.
GTK_SCROLL_PAGE_FORWARD	Page forward.
GTK_SCROLL_STEP_UP	Step up.
GTK_SCROLL_STEP_DOWN	Step down.
GTK_SCROLL_PAGE_UP	Page up.
GTK_SCROLL_PAGE_DOWN	Page down.
GTK_SCROLL_STEP_LEFT	Step to the left.
GTK_SCROLL_STEP_RIGHT	Step to the right.
GTK_SCROLL_PAGE_LEFT	Page to the left.
GTK_SCROLL_PAGE_RIGHT	Page to the right.
GTK_SCROLL_START	Scroll to start.
GTK_SCROLL_END	Scroll to end.

enum GtkSelectionMode

Used to control what selections users are allowed to make.

Members

GTK_SELECTION_NONE	No selection is possible.
GTK_SELECTION_SINGLE	Zero or one element may be selected.
GTK_SELECTION_BROWSE	Exactly one element is selected. In some circumstances, such as initially or during a search operation, it's possible for no element to be selected with `GTK_SELECTION_BROWSE`. What is really enforced is that the user can't deselect a currently selected element except by selecting another element.
GTK_SELECTION_MULTIPLE	Any number of elements may be selected. The Ctrl key may be used to enlarge the selection, and Shift key to select between the focus and the child pointed to. Some widgets may also allow Click-drag to select a range of elements.

---

enum GtkShadowType

Used to change the appearance of an outline typically provided by a GtkFrame.

Note that many themes do not differentiate the appearance of the various shadow types: Either their is no visible shadow (`GTK_SHADOW_NONE`), or there is (any other value).

Members

GTK_SHADOW_NONE	No outline.
GTK_SHADOW_IN	The outline is bevelled inwards.
GTK_SHADOW_OUT	The outline is bevelled outwards like a button.
GTK_SHADOW_ETCHED_IN	The outline has a sunken 3d appearance.
GTK_SHADOW_ETCHED_OUT	The outline has a raised 3d appearance.

---

enum GtkStateFlags

Describes a widget state. Widget states are used to match the widget against CSS pseudo-classes. Note that GTK extends the regular CSS classes and sometimes uses different names.

Members

GTK_STATE_FLAG_NORMAL	State during normal operation.
GTK_STATE_FLAG_ACTIVE	Widget is active.
GTK_STATE_FLAG_PRELIGHT	Widget has a mouse pointer over it.
GTK_STATE_FLAG_SELECTED	Widget is selected.

GTK_STATE_FLAG_INSENSITIVE	Widget is insensitive.
GTK_STATE_FLAG_INCONSISTENT	Widget is inconsistent.
GTK_STATE_FLAG_FOCUSED	Widget has the keyboard focus.
GTK_STATE_FLAG_BACKDROP	Widget is in a background toplevel window.
GTK_STATE_FLAG_DIR_LTR	Widget is in left-to-right text direction. Since 3.8
GTK_STATE_FLAG_DIR_RTL	Widget is in right-to-left text direction. Since 3.8
GTK_STATE_FLAG_LINK	Widget is a link. Since 3.12
GTK_STATE_FLAG_VISITED	The location the widget points to has already been visited. Since 3.12
GTK_STATE_FLAG_CHECKED	Widget is checked. Since 3.14
GTK_STATE_FLAG_DROP_ACTIVE	Widget is highlighted as a drop target for DND. Since 3.20

enum GtkToolbarStyle

Used to customize the appearance of a GtkToolbar. Note that setting the toolbar style overrides the user's preferences for the default toolbar style. Note that if the button has only a label set and GTK_TOOLBAR_ICONS is used, the label will be visible, and vice versa.

Members

GTK_TOOLBAR_ICONS	Buttons display only icons in the toolbar.
GTK_TOOLBAR_TEXT	Buttons display only text labels in the toolbar.
GTK_TOOLBAR_BOTH	Buttons display text and icons in the toolbar.
GTK_TOOLBAR_BOTH_HORIZ	Buttons display icons and text alongside each other, rather than vertically stacked

enum GtkSortType

Determines the direction of a sort.

Members

GTK_SORT_ASCENDING	Sorting is in ascending order.
GTK_SORT_DESCENDING	Sorting is in descending order.

Selections

Selections — Functions for handling inter-process communication via selections

Functions

GtkTargetEntry *	gtk_target_entry_new ()
GtkTargetEntry *	gtk_target_entry_copy ()
void	gtk_target_entry_free ()
GtkTargetList *	gtk_target_list_new ()
GtkTargetList *	gtk_target_list_ref ()
void	gtk_target_list_unref ()
void	gtk_target_list_add ()
void	gtk_target_list_add_table ()
void	gtk_target_list_add_text_targets ()
void	gtk_target_list_add_image_targets ()
void	gtk_target_list_add_uri_targets ()
void	gtk_target_list_add_rich_text_targets ()
void	gtk_target_list_remove ()
gboolean	gtk_target_list_find ()
void	gtk_target_table_free ()
GtkTargetEntry *	gtk_target_table_new_from_list ()
gboolean	gtk_selection_owner_set ()
gboolean	gtk_selection_owner_set_for_display ()
void	gtk_selection_add_target ()
void	gtk_selection_add_targets ()
void	gtk_selection_clear_targets ()
gboolean	gtk_selection_convert ()
void	gtk_selection_data_set ()
gboolean	gtk_selection_data_set_text ()
guchar *	gtk_selection_data_get_text ()
gboolean	gtk_selection_data_set_pixbuf ()
GdkPixbuf *	gtk_selection_data_get_pixbuf ()
gboolean	gtk_selection_data_set_uris ()
gchar **	gtk_selection_data_get_uris ()
gboolean	gtk_selection_data_get_targets ()
gboolean	gtk_selection_data_targets_include_image ()
gboolean	gtk_selection_data_targets_include_text ()
gboolean	gtk_selection_data_targets_include_uri ()
gboolean	gtk_selection_data_targets_include_rich_text ()
GdkAtom	gtk_selection_data_get_selection ()
const guchar *	gtk_selection_data_get_data ()
gint	gtk_selection_data_get_length ()
const guchar *	gtk_selection_data_get_data_with_length ()
GdkAtom	gtk_selection_data_get_data_type ()
GdkDisplay *	gtk_selection_data_get_display ()
gint	gtk_selection_data_get_format ()
GdkAtom	gtk_selection_data_get_target ()
gboolean	gtk_targets_include_image ()
gboolean	gtk_targets_include_text ()
gboolean	gtk_targets_include_uri ()
gboolean	gtk_targets_include_rich_text ()
void	gtk_selection_remove_all ()
GtkSelectionData *	gtk_selection_data_copy ()

void	gtk_selection_data_free ()

Types and Values

	GtkSelectionData
struct	GtkTargetEntry
	GtkTargetList
struct	GtkTargetPair

Object Hierarchy

```
GBoxed
├── GtkSelectionData
└── GtkTargetList
```

Includes

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

Description

The selection mechanism provides the basis for different types of communication between processes. In particular, drag and drop and GtkClipboard work via selections. You will very seldom or never need to use most of the functions in this section directly; GtkClipboard provides a nicer interface to the same functionality.

If an application is expected to exchange image data and work on Windows, it is highly advised to support at least "image/bmp" target for the widest possible compatibility with third-party applications. GtkClipboard already does that by using `gtk_target_list_add_image_targets()` and `gtk_selection_data_set_pixbuf()` or `gtk_selection_data_get_pixbuf()`, which is one of the reasons why it is advised to use GtkClipboard.

Some of the datatypes defined this section are used in the GtkClipboard and drag-and-drop API's as well. The GtkTargetEntry and GtkTargetList objects represent lists of data types that are supported when sending or receiving data. The GtkSelectionData object is used to store a chunk of data along with the data type and other associated information.

Functions

gtk_target_entry_new ()

```
GtkTargetEntry *
gtk_target_entry_new (const gchar *target,
                      guint flags,
                      guint info);
```

Makes a new GtkTargetEntry.

Parameters

target	String identifier for target
flags	Set of flags, see GtkTargetFlags
info	an ID that will be passed back to the application

Returns

a pointer to a new GtkTargetEntry. Free with `gtk_target_entry_free()`

---

gtk_target_entry_copy ()

```
GtkTargetEntry *
gtk_target_entry_copy (GtkTargetEntry *data);
```

Makes a copy of a GtkTargetEntry and its data.

Parameters

data	a pointer to a GtkTargetEntry

Returns

a pointer to a copy of `data` . Free with `gtk_target_entry_free()`

---

gtk_target_entry_free ()

```
void
gtk_target_entry_free (GtkTargetEntry *data);
```

Frees a GtkTargetEntry returned from `gtk_target_entry_new()` or `gtk_target_entry_copy()`.

Parameters

data	a pointer to a GtkTargetEntry.

---

gtk_target_list_new ()

```
GtkTargetList *
gtk_target_list_new (const GtkTargetEntry *targets,
                     guint ntargets);
```

Creates a new GtkTargetList from an array of GtkTargetEntry.

Parameters

targets	Pointer to an array of GtkTargetEntry.	[array length=ntargets][allow-none]
ntargets	number of entries in `targets` .	

Returns

the new GtkTargetList.

[transfer full]

---

gtk_target_list_ref ()

```
GtkTargetList *
gtk_target_list_ref (GtkTargetList *list);
```

Increases the reference count of a GtkTargetList by one.

Parameters

list	a GtkTargetList

Returns

the passed in GtkTargetList.

---

gtk_target_list_unref ()

```
void
gtk_target_list_unref (GtkTargetList *list);
```

Decreases the reference count of a GtkTargetList by one. If the resulting reference count is zero, frees the list.

Parameters

list	a GtkTargetList

---

gtk_target_list_add ()

```
void
gtk_target_list_add (GtkTargetList *list,
                     GdkAtom target,
                     guint flags,
```

```
                        guint info);
```
Appends another target to a GtkTargetList.

Parameters

list	a GtkTargetList	
target	the interned atom representing the target	
flags	the flags for this target	
info	an ID that will be passed back to the application	

---

gtk_target_list_add_table ()

```
void
gtk_target_list_add_table (GtkTargetList *list,
                           const GtkTargetEntry *targets,
                           guint ntargets);
```
Prepends a table of GtkTargetEntry to a target list.

Parameters

list	a GtkTargetList	
targets	the table of GtkTargetEntry.	[array length=ntargets]
ntargets	number of targets in the table	

---

gtk_target_list_add_text_targets ()

```
void
gtk_target_list_add_text_targets (GtkTargetList *list,
                                  guint info);
```
Appends the text targets supported by GtkSelectionData to the target list. All targets are added with the same `info`.

Parameters

list	a GtkTargetList
info	an ID that will be passed back to the application

Since: 2.6

---

gtk_target_list_add_image_targets ()

```
void
```

```
gtk_target_list_add_image_targets (GtkTargetList *list,
                                   guint info,
                                   gboolean writable);
```

Appends the image targets supported by GtkSelectionData to the target list. All targets are added with the same `info`.

Parameters

list	a GtkTargetList
info	an ID that will be passed back to the application
writable	whether to add only targets for which GTK+ knows how to convert a pixbuf into the format

Since: 2.6

---

gtk_target_list_add_uri_targets ()

```
void
gtk_target_list_add_uri_targets (GtkTargetList *list,
                                 guint info);
```

Appends the URI targets supported by GtkSelectionData to the target list. All targets are added with the same `info`.

Parameters

list	a GtkTargetList
info	an ID that will be passed back to the application

Since: 2.6

---

gtk_target_list_add_rich_text_targets ()

```
void
gtk_target_list_add_rich_text_targets (GtkTargetList *list,
                                       guint info,
                                       gboolean deserializable,
                                       GtkTextBuffer *buffer);
```

Appends the rich text targets registered with `gtk_text_buffer_register_serialize_format()` or `gtk_text_buffer_register_deserialize_format()` to the target list. All targets are added with the same `info`.

Parameters

list	a GtkTargetList
info	an ID that will be passed back to the application
deserializable	if `TRUE`, then deserializable rich text formats will be added, serializable formats otherwise.
buffer	a GtkTextBuffer.

Since: 2.10

gtk_target_list_remove ()

```
void
gtk_target_list_remove (GtkTargetList *list,
                        GdkAtom target);
```

Removes a target from a target list.

Parameters

list	a GtkTargetList
target	the interned atom representing the target

gtk_target_list_find ()

```
gboolean
gtk_target_list_find (GtkTargetList *list,
                      GdkAtom target,
                      guint *info);
```

Looks up a given target in a GtkTargetList.

Parameters

list	a GtkTargetList	
target	an interned atom representing the target to search for	
info	a pointer to the location to store application info for target, or NULL.	[out][allow-none]

Returns

TRUE if the target was found, otherwise FALSE

gtk_target_table_free ()

```
void
gtk_target_table_free (GtkTargetEntry *targets,
                       gint n_targets);
```

This function frees a target table as returned by gtk_target_table_new_from_list()

Parameters

targets	a GtkTargetEntry array.	[array length=n_targets]

n_targets	the number of entries in the array

Since: 2.10

---

gtk_target_table_new_from_list ()

```
GtkTargetEntry *
gtk_target_table_new_from_list (GtkTargetList *list,
                                gint *n_targets);
```

This function creates an GtkTargetEntry array that contains the same targets as the passed `list`. The returned table is newly allocated and should be freed using `gtk_target_table_free()` when no longer needed.

Parameters

list	a GtkTargetList	
n_targets	return location for the number ot targets in the table.	[out]

Returns

the new table.

[array length=n_targets][transfer full]

Since: 2.10

---

gtk_selection_owner_set ()

```
gboolean
gtk_selection_owner_set (GtkWidget *widget,
                         GdkAtom selection,
                         guint32 time_);
```

Claims ownership of a given selection for a particular widget, or, if `widget` is `NULL`, release ownership of the selection.

Parameters

widget	a GtkWidget, or `NULL`.	[allow-none]
selection	an interned atom representing the selection to claim	
time_	timestamp with which to claim the selection	

Returns

`TRUE` if the operation succeeded

---

gtk_selection_owner_set_for_display ()

```
gboolean
gtk_selection_owner_set_for_display (GdkDisplay *display,
                                     GtkWidget *widget,
                                     GdkAtom selection,
                                     guint32 time_);
```

Claim ownership of a given selection for a particular widget, or, if `widget` is `NULL`, release ownership of the selection.

Parameters

display	the GdkDisplay where the selection is set	
widget	new selection owner (a GtkWidget), or `NULL`.	[allow-none]
selection	an interned atom representing the selection to claim.	
time_	timestamp with which to claim the selection	

Returns

TRUE if the operation succeeded

Since: 2.2

---

gtk_selection_add_target ()

```
void
gtk_selection_add_target (GtkWidget *widget,
                          GdkAtom selection,
                          GdkAtom target,
                          guint info);
```

Appends a specified target to the list of supported targets for a given widget and selection.

Parameters

widget	a GtkWidget
selection	the selection
target	target to add.
info	A unsigned integer which will be passed back to the application.

---

gtk_selection_add_targets ()

```
void
gtk_selection_add_targets (GtkWidget *widget,
                           GdkAtom selection,
                           const GtkTargetEntry *targets,
                           guint ntargets);
```

Prepends a table of targets to the list of supported targets for a given widget and selection.

Parameters

widget	a GtkWidget	
selection	the selection	
targets	a table of targets to add.	[array length=ntargets]
ntargets	number of entries in `targets`	

---

gtk_selection_clear_targets ()

```
void
gtk_selection_clear_targets (GtkWidget *widget,
                             GdkAtom selection);
```

Remove all targets registered for the given selection for the widget.

Parameters

widget	a GtkWidget
selection	an atom representing a selection

---

gtk_selection_convert ()

```
gboolean
gtk_selection_convert (GtkWidget *widget,
                       GdkAtom selection,
                       GdkAtom target,
                       guint32 time_);
```

Requests the contents of a selection. When received, a “selection-received” signal will be generated.

Parameters

widget	The widget which acts as requestor
selection	Which selection to get
target	Form of information desired (e.g., STRING)
time_	Time of request (usually of triggering event) In emergency, you could use GDK_CURRENT_TIME

Returns

`TRUE` if requested succeeded. `FALSE` if we could not process request. (e.g., there was already a request in process for this widget).

---

gtk_selection_data_set ()

```
void
gtk_selection_data_set (GtkSelectionData *selection_data,
                        GdkAtom type,
                        gint format,
                        const guchar *data,
                        gint length);
```

Stores new data into a GtkSelectionData object. Should only be called from a selection handler callback. Zero-terminates the stored data.

Parameters

selection_data	a pointer to a GtkSelectionData.	
type	the type of selection data	
format	format (number of bits in a unit)	
data	pointer to the data (will be copied).	[array length=length]
length	length of the data	

---

gtk_selection_data_set_text ()

```
gboolean
gtk_selection_data_set_text (GtkSelectionData *selection_data,
                             const gchar *str,
                             gint len);
```

Sets the contents of the selection from a UTF-8 encoded string. The string is converted to the form determined by `selection_data->target` .

Parameters

selection_data	a GtkSelectionData
str	a UTF-8 string
len	the length of `str` , or -1 if `str` is nul-terminated.

Returns

`TRUE` if the selection was successfully set, otherwise `FALSE`.

---

gtk_selection_data_get_text ()

```
guchar *
gtk_selection_data_get_text (const GtkSelectionData *selection_data);
```

Gets the contents of the selection data as a UTF-8 string.

Parameters

selection_data	a GtkSelectionData

Returns

if the selection data contained a recognized text type and it could be converted to UTF-8, a newly allocated string containing the converted text, otherwise `NULL`. If the result is non-`NULL` it must be freed with `g_free()`.

[type utf8][nullable][transfer full]

---

gtk_selection_data_set_pixbuf ()

```
gboolean
gtk_selection_data_set_pixbuf (GtkSelectionData *selection_data,
                               GdkPixbuf *pixbuf);
```

Sets the contents of the selection from a GdkPixbuf The pixbuf is converted to the form determined by `selection_data->target`.

Parameters

selection_data	a GtkSelectionData
pixbuf	a GdkPixbuf

Returns

`TRUE` if the selection was successfully set, otherwise `FALSE`.

Since: 2.6

---

gtk_selection_data_get_pixbuf ()

```
GdkPixbuf *
gtk_selection_data_get_pixbuf (const GtkSelectionData *selection_data);
```

Gets the contents of the selection data as a GdkPixbuf.

Parameters

selection_data	a GtkSelectionData

Returns

if the selection data contained a recognized image type and it could be converted to a GdkPixbuf, a newly allocated pixbuf is returned, otherwise `NULL`. If the result is non-`NULL` it must be freed with `g_object_unref()`.

[nullable][transfer full]

Since: 2.6

---

gtk_selection_data_set_uris ()

```
gboolean
gtk_selection_data_set_uris (GtkSelectionData *selection_data,
                             gchar **uris);
```

Sets the contents of the selection from a list of URIs. The string is converted to the form determined by `selection_data->target`.

Parameters

selection_data	a GtkSelectionData	
uris	a `NULL`-terminated array of strings holding URIs.	[array zero-terminated=1]

Returns

`TRUE` if the selection was successfully set, otherwise `FALSE`.

Since: 2.6

---

gtk_selection_data_get_uris ()

```
gchar **
gtk_selection_data_get_uris (const GtkSelectionData *selection_data);
```

Gets the contents of the selection data as array of URIs.

Parameters

selection_data	a GtkSelectionData

Returns

if the selection data contains a list of URIs, a newly allocated `NULL`-terminated string array containing the URIs, otherwise `NULL`. If the result is non-`NULL` it must be freed with `g_strfreev()`.

[array zero-terminated=1][element-type utf8][transfer full]

Since: 2.6

---

gtk_selection_data_get_targets ()

```
gboolean
gtk_selection_data_get_targets (const GtkSelectionData *selection_data,
                                GdkAtom **targets,
                                gint *n_atoms);
```

Gets the contents of `selection_data` as an array of targets. This can be used to interpret the results of getting the standard TARGETS target that is always supplied for any selection.

Parameters

selection_data	a GtkSelectionData object	
targets	location to store an array of targets. The result stored here must be freed with `g_free()`.	[out][array length=n_atoms] [transfer container]
n_atoms	location to store number of items in `targets` .	

Returns

`TRUE` if `selection_data` contains a valid array of targets, otherwise `FALSE`.

---

gtk_selection_data_targets_include_image ()

```
gboolean
gtk_selection_data_targets_include_image
                                (const GtkSelectionData *selection_data,
                                 gboolean writable);
```

Given a GtkSelectionData object holding a list of targets, determines if any of the targets in `targets` can be used to provide a GdkPixbuf.

Parameters

selection_data	a GtkSelectionData object
writable	whether to accept only targets for which GTK+ knows how to convert a pixbuf into the format

Returns

`TRUE` if `selection_data` holds a list of targets, and a suitable target for images is included, otherwise `FALSE`.

Since: 2.6

---

gtk_selection_data_targets_include_text ()

```
gboolean
gtk_selection_data_targets_include_text
                                (const GtkSelectionData *selection_data);
```

Given a GtkSelectionData object holding a list of targets, determines if any of the targets in `targets` can be

used to provide text.

Parameters

selection_data	a GtkSelectionData object

Returns

TRUE if selection_data holds a list of targets, and a suitable target for text is included, otherwise FALSE.

---

gtk_selection_data_targets_include_uri ()

```
gboolean
gtk_selection_data_targets_include_uri
                                (const GtkSelectionData *selection_data);
```

Given a GtkSelectionData object holding a list of targets, determines if any of the targets in targets can be used to provide a list or URIs.

Parameters

selection_data	a GtkSelectionData object

Returns

TRUE if selection_data holds a list of targets, and a suitable target for URI lists is included, otherwise FALSE.

Since: 2.10

---

gtk_selection_data_targets_include_rich_text ()

```
gboolean
gtk_selection_data_targets_include_rich_text
                                (const GtkSelectionData *selection_data,
                                 GtkTextBuffer *buffer);
```

Given a GtkSelectionData object holding a list of targets, determines if any of the targets in targets can be used to provide rich text.

Parameters

selection_data	a GtkSelectionData object
buffer	a GtkTextBuffer

Returns

`TRUE` if `selection_data` holds a list of targets, and a suitable target for rich text is included, otherwise `FALSE`.

Since: 2.10

---

gtk_selection_data_get_selection ()

```
GdkAtom
gtk_selection_data_get_selection (const GtkSelectionData *selection_data);
```

Retrieves the selection GdkAtom of the selection data.

Parameters

selection_data	a pointer to a GtkSelectionData.

Returns

the selection GdkAtom of the selection data.

[transfer none]

Since: 2.16

---

gtk_selection_data_get_data ()

```
const guchar *
gtk_selection_data_get_data (const GtkSelectionData *selection_data);
```

Retrieves the raw data of the selection.

[skip]

Parameters

selection_data	a pointer to a GtkSelectionData.

Returns

the raw data of the selection.

[array][element-type guint8]

Since: 2.14

---

gtk_selection_data_get_length ()

```
gint
gtk_selection_data_get_length (const GtkSelectionData *selection_data);
```

Retrieves the length of the raw data of the selection.

Parameters

selection_data	a pointer to a GtkSelectionData.

Returns

the length of the data of the selection.

Since: 2.14

---

gtk_selection_data_get_data_with_length ()

```
const guchar *
gtk_selection_data_get_data_with_length
                                (const GtkSelectionData *selection_data,
                                 gint *length);
```

Retrieves the raw data of the selection along with its length.

[rename-to gtk_selection_data_get_data]

Parameters

selection_data	a pointer to a GtkSelectionData.	
length	return location for length of the data segment.	[out]

Returns

the raw data of the selection.

[array length=length]

Since: 3.0

---

gtk_selection_data_get_data_type ()

```
GdkAtom
gtk_selection_data_get_data_type (const GtkSelectionData *selection_data);
```

Retrieves the data type of the selection.

Parameters

selection_data	a pointer to a GtkSelectionData.

Returns

the data type of the selection.

[transfer none]

Since: 2.14

---

gtk_selection_data_get_display ()

```
GdkDisplay *
gtk_selection_data_get_display (const GtkSelectionData *selection_data);
```

Retrieves the display of the selection.

Parameters

selection_data	a pointer to a GtkSelectionData.

Returns

the display of the selection.

[transfer none]

Since: 2.14

---

gtk_selection_data_get_format ()

```
gint
gtk_selection_data_get_format (const GtkSelectionData *selection_data);
```

Retrieves the format of the selection.

Parameters

selection_data	a pointer to a GtkSelectionData.

Returns

the format of the selection.

Since: 2.14

---

gtk_selection_data_get_target ()

```
GdkAtom
gtk_selection_data_get_target (const GtkSelectionData *selection_data);
```

Retrieves the target of the selection.

Parameters

selection_data	a pointer to a GtkSelectionData.	

Returns

the target of the selection.

[transfer none]

Since: 2.14

---

gtk_targets_include_image ()

```
gboolean
gtk_targets_include_image (GdkAtom *targets,
                           gint n_targets,
                           gboolean writable);
```

Determines if any of the targets in `targets` can be used to provide a GdkPixbuf.

Parameters

targets	an array of GdkAtoms.	[array length=n_targets]
n_targets	the length of `targets`	
writable	whether to accept only targets for which GTK+ knows how to convert a pixbuf into the format	

Returns

`TRUE` if `targets` include a suitable target for images, otherwise `FALSE`.

Since: 2.10

---

gtk_targets_include_text ()

```
gboolean
gtk_targets_include_text (GdkAtom *targets,
                          gint n_targets);
```

Determines if any of the targets in `targets` can be used to provide text.

Parameters

targets	an array of GdkAtoms.	[array length=n_targets]
n_targets	the length of `targets`	

Returns

`TRUE` if `targets` include a suitable target for text, otherwise `FALSE`.

Since: 2.10

---

gtk_targets_include_uri ()

```
gboolean
gtk_targets_include_uri (GdkAtom *targets,
                         gint n_targets);
```

Determines if any of the targets in `targets` can be used to provide an uri list.

Parameters

targets	an array of GdkAtoms.	[array length=n_targets]
n_targets	the length of `targets`	

Returns

`TRUE` if `targets` include a suitable target for uri lists, otherwise `FALSE`.

Since: 2.10

---

gtk_targets_include_rich_text ()

```
gboolean
gtk_targets_include_rich_text (GdkAtom *targets,
                               gint n_targets,
                               GtkTextBuffer *buffer);
```

Determines if any of the targets in `targets` can be used to provide rich text.

Parameters

targets	an array of GdkAtoms.	[array length=n_targets]
n_targets	the length of `targets`	
buffer	a GtkTextBuffer	

Returns

`TRUE` if `targets` include a suitable target for rich text, otherwise `FALSE`.

Since: 2.10

---

gtk_selection_remove_all ()

```
void
gtk_selection_remove_all (GtkWidget *widget);
```

Removes all handlers and unsets ownership of all selections for a widget. Called when widget is being destroyed. This function will not generally be called by applications.

Parameters

widget	a GtkWidget

---

gtk_selection_data_copy ()

```
GtkSelectionData *
gtk_selection_data_copy (const GtkSelectionData *data);
```

Makes a copy of a GtkSelectionData and its data.

Parameters

data	a pointer to a GtkSelectionData.

Returns

a pointer to a copy of `data` .

---

gtk_selection_data_free ()

```
void
gtk_selection_data_free (GtkSelectionData *data);
```

Frees a GtkSelectionData returned from `gtk_selection_data_copy()`.

Parameters

data	a pointer to a GtkSelectionData.

Types and Values

GtkSelectionData

```
typedef struct _GtkSelectionData GtkSelectionData;
```

---

struct GtkTargetEntry

```
struct GtkTargetEntry {
  gchar *target;
  guint  flags;
  guint  info;
};
```

A GtkTargetEntry represents a single type of data than can be supplied for by a widget for a selection or for supplied or received during drag-and-drop.

Members

gchar *`target`;	a string representation of the target type
guint `flags`;	GtkTargetFlags for DND
guint `info`;	an application-assigned integer ID which will get passed as a parameter to e.g the "selection-get" signal. It allows the application to identify the target type without extensive string compares.

GtkTargetList

```
typedef struct _GtkTargetList GtkTargetList;
```

A GtkTargetList is a reference counted list of GtkTargetPair and should be treated as opaque.

---

struct GtkTargetPair

```
struct GtkTargetPair {
  GdkAtom   target;
  guint     flags;
  guint     info;
};
```

A GtkTargetPair is used to represent the same information as a table of GtkTargetEntry, but in an efficient form.

Members

GdkAtom `target`;	GdkAtom representation of the target type

guint `flags`;	GtkTargetFlags for DND
guint `info`;	an application-assigned integer ID which will get passed as a parameter to e.g the "selection-get" signal. It allows the application to identify the target type without extensive string compares.

See Also

GtkWidget - Much of the operation of selections happens via signals for GtkWidget. In particular, if you are using the functions in this section, you may need to pay attention to "selection-get", "selection-received" and "selection-clear-event" signals

---

Testing

Testing — Utilities for testing GTK+ applications

Functions

GtkWidget *	gtk_test_create_simple_window ()
GtkWidget *	gtk_test_create_widget ()
GtkWidget *	gtk_test_display_button_window ()
GtkWidget *	gtk_test_find_label ()
GtkWidget *	gtk_test_find_sibling ()
GtkWidget *	gtk_test_find_widget ()
void	gtk_test_init ()
const GType *	gtk_test_list_all_types ()
void	gtk_test_register_all_types ()
double	gtk_test_slider_get_value ()
void	gtk_test_slider_set_perc ()
gboolean	gtk_test_spin_button_click ()
gchar *	gtk_test_text_get ()
void	gtk_test_text_set ()
gboolean	gtk_test_widget_click ()
gboolean	gtk_test_widget_send_key ()
void	gtk_test_widget_wait_for_draw ()

Includes

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

Description

Functions

gtk_test_create_simple_window ()

```
GtkWidget *
gtk_test_create_simple_window (const gchar *window_title,
                               const gchar *dialog_text);
```

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

This testing infrastructure is phased out in favor of reftests.

Create a simple window with window title `window_title` and text contents `dialog_text` . The window will quit any running `gtk_main()`-loop when destroyed, and it will automatically be destroyed upon test function teardown.

Parameters

window_title	Title of the window to be displayed.
dialog_text	Text inside the window to be displayed.

Returns

a widget pointer to the newly created GtkWindow.

[transfer none]

Since: 2.14

---

gtk_test_create_widget ()

```
GtkWidget *
gtk_test_create_widget (GType widget_type,
                        const gchar *first_property_name,
                        ...);
```

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

This testing infrastructure is phased out in favor of reftests.

This function wraps `g_object_new()` for widget types. It'll automatically show all created non window widgets, also `g_object_ref_sink()` them (to keep them alive across a running test) and set them up for destruction during the next test teardown phase.

Parameters

widget_type	a valid widget type.	
first_property_name	Name of first property to set or `NULL`.	[allow-none]
...	value to set the first property to, followed by more name-value pairs, terminated by `NULL`	

Returns

a newly created widget.

[transfer none]

Since: 2.14

---

gtk_test_display_button_window ()

```
GtkWidget *
gtk_test_display_button_window (const gchar *window_title,
                                const gchar *dialog_text,
                                ...);
```

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

This testing infrastructure is phased out in favor of reftests.

Create a window with window title `window_title` , text contents `dialog_text` , and a number of buttons, according to the paired argument list given as @... parameters. Each button is created with a `label` and a ::clicked signal handler that incremrents the integer stored in `nump` . The window will be automatically shown with `gtk_widget_show_now()` after creation, so when this function returns it has already been mapped, resized and positioned on screen. The window will quit any running `gtk_main()`-loop when destroyed, and it will automatically be destroyed upon test function teardown.

Parameters

window_title	Title of the window to be displayed.
dialog_text	Text inside the window to be displayed.
...	`NULL` terminated list of (const char *label, int *nump) pairs.

Returns

a widget pointer to the newly created GtkWindow.

[transfer full]

Since: 2.14

---

gtk_test_find_label ()

```
GtkWidget *
gtk_test_find_label (GtkWidget *widget,
                     const gchar *label_pattern);
```

This function will search `widget` and all its descendants for a GtkLabel widget with a text string matching `label_pattern` . The `label_pattern` may contain asterisks “*” and question marks “?” as placeholders, `g_pattern_match()` is used for the matching. Note that locales other than "C“ tend to alter (translate” label

strings, so this function is genrally only useful in test programs with predetermined locales, see gtk_test_init() for more details.

Parameters

widget	Valid label or container widget.
label_pattern	Shell-glob pattern to match a label string.

Returns

a GtkLabel widget if any is found.

[transfer none]

Since: 2.14

---

gtk_test_find_sibling ()

```
GtkWidget *
gtk_test_find_sibling (GtkWidget *base_widget,
                       GType widget_type);
```

This function will search siblings of `base_widget` and siblings of its ancestors for all widgets matching `widget_type` . Of the matching widgets, the one that is geometrically closest to `base_widget` will be returned. The general purpose of this function is to find the most likely “action” widget, relative to another labeling widget. Such as finding a button or text entry widget, given its corresponding label widget.

Parameters

base_widget	Valid widget, part of a widget hierarchy
widget_type	Type of a aearched for sibling widget

Returns

a widget of type `widget_type` if any is found.

[transfer none]

Since: 2.14

---

gtk_test_find_widget ()

```
GtkWidget *
gtk_test_find_widget (GtkWidget *widget,
                      const gchar *label_pattern,
                      GType widget_type);
```

This function will search the descendants of `widget` for a widget of type `widget_type` that has a label matching

`label_pattern` next to it. This is most useful for automated GUI testing, e.g. to find the “OK” button in a dialog and synthesize clicks on it. However see `gtk_test_find_label()`, `gtk_test_find_sibling()` and `gtk_test_widget_click()` for possible caveats involving the search of such widgets and synthesizing widget events.

Parameters

widget	Container widget, usually a GtkWindow.
label_pattern	Shell-glob pattern to match a label string.
widget_type	Type of a aearched for label sibling widget.

Returns

a valid widget if any is found or `NULL`.

[nullable][transfer none]

Since: 2.14

---

gtk_test_init ()

```
void
gtk_test_init (int *argcp,
               char ***argvp,
               ...);
```

This function is used to initialize a GTK+ test program.

It will in turn call `g_test_init()` and `gtk_init()` to properly initialize the testing framework and graphical toolkit. It’ll also set the program’s locale to “C” and prevent loading of rc files and Gtk+ modules. This is done to make tets program environments as deterministic as possible.

Like `gtk_init()` and `g_test_init()`, any known arguments will be processed and stripped from `argc` and `argv`.

Parameters

argcp	Address of the `argc` parameter of the `main()` function. Changed if any arguments were handled.	
argvp	Address of the `argv` parameter of `main()`. Any parameters understood by `g_test_init()` or `gtk_init()` are stripped before return.	[inout][array length=argcp]
...	currently unused	

Since: 2.14

---

gtk_test_list_all_types ()

```
const GType *
gtk_test_list_all_types (guint *n_types);
```

Return the type ids that have been registered after calling `gtk_test_register_all_types()`.

Parameters

n_types	location to store number of types

Returns

0-terminated array of type ids.

[array length=n_types zero-terminated=1][transfer none]

Since: 2.14

---

gtk_test_register_all_types ()

```
void
gtk_test_register_all_types (void);
```

Force registration of all core Gtk+ and Gdk object types. This allowes to refer to any of those object types via `g_type_from_name()` after calling this function.

Since: 2.14

---

gtk_test_slider_get_value ()

```
double
gtk_test_slider_get_value (GtkWidget *widget);
```

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

This testing infrastructure is phased out in favor of reftests.

Retrive the literal adjustment value for GtkRange based widgets and spin buttons. Note that the value returned by this function is anything between the lower and upper bounds of the adjustment belonging to `widget` , and is not a percentage as passed in to `gtk_test_slider_set_perc()`.

Parameters

widget	valid widget pointer.

Returns

gtk_adjustment_get_value (adjustment) for an adjustment belonging to `widget` .

Since: 2.14

---

gtk_test_slider_set_perc ()

```
void
gtk_test_slider_set_perc (GtkWidget *widget,
                          double percentage);
```

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

This testing infrastructure is phased out in favor of reftests.

This function will adjust the slider position of all GtkRange based widgets, such as scrollbars or scales, it'll also adjust spin buttons. The adjustment value of these widgets is set to a value between the lower and upper limits, according to the `percentage` argument.

Parameters

widget	valid widget pointer.
percentage	value between 0 and 100.

Since: 2.14

---

gtk_test_spin_button_click ()

```
gboolean
gtk_test_spin_button_click (GtkSpinButton *spinner,
                            guint button,
                            gboolean upwards);
```

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

This testing infrastructure is phased out in favor of reftests.

This function will generate a `button` click in the upwards or downwards spin button arrow areas, usually leading to an increase or decrease of spin button's value.

Parameters

spinner	valid GtkSpinButton widget.
button	Number of the pointer button for the event, usually 1, 2 or 3.
upwards	`TRUE` for upwards arrow click, `FALSE` for downwards arrow click.

Returns

whether all actions neccessary for the button click simulation were carried out successfully.

Since: 2.14

---

gtk_test_text_get ()

```
gchar *
gtk_test_text_get (GtkWidget *widget);
```

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

This testing infrastructure is phased out in favor of reftests.

Retrive the text string of `widget` if it is a GtkLabel, GtkEditable (entry and text widgets) or GtkTextView.

Parameters

widget	valid widget pointer.

Returns

new 0-terminated C string, needs to be released with `g_free()`.

Since: 2.14

---

gtk_test_text_set ()

```
void
gtk_test_text_set (GtkWidget *widget,
                   const gchar *string);
```

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

This testing infrastructure is phased out in favor of reftests.

Set the text string of `widget` to `string` if it is a GtkLabel, GtkEditable (entry and text widgets) or GtkTextView.

Parameters

widget	valid widget pointer.
string	a 0-terminated C string

Since: 2.14

---

gtk_test_widget_click ()

```
gboolean
gtk_test_widget_click (GtkWidget *widget,
                       guint button,
                       GdkModifierType modifiers);
```

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

This testing infrastructure is phased out in favor of reftests.

This function will generate a `button` click (button press and button release event) in the middle of the first GdkWindow found that belongs to `widget` . For windowless widgets like GtkButton (which returns `FALSE` from `gtk_widget_get_has_window()`), this will often be an input-only event window. For other widgets, this is usually widget->window. Certain caveats should be considered when using this function, in particular because the mouse pointer is warped to the button click location, see `gdk_test_simulate_button()` for details.

Parameters

widget	Widget to generate a button click on.
button	Number of the pointer button for the event, usually 1, 2 or 3.
modifiers	Keyboard modifiers the event is setup with.

Returns

whether all actions neccessary for the button click simulation were carried out successfully.

Since: 2.14

---

gtk_test_widget_send_key ()

```
gboolean
gtk_test_widget_send_key (GtkWidget *widget,
                          guint keyval,
                          GdkModifierType modifiers);
```

This function will generate keyboard press and release events in the middle of the first GdkWindow found that belongs to `widget` . For windowless widgets like GtkButton (which returns `FALSE` from `gtk_widget_get_has_window()`), this will often be an input-only event window. For other widgets, this is usually widget->window. Certain caveats should be considered when using this function, in particular because the mouse pointer is warped to the key press location, see `gdk_test_simulate_key()` for details.

Parameters

widget	Widget to generate a key press and release on.
keyval	A Gdk keyboard value.
modifiers	Keyboard modifiers the event is setup with.

Returns

whether all actions neccessary for the key event simulation were carried out successfully.

Since: 2.14

---

gtk_test_widget_wait_for_draw ()

```
void
gtk_test_widget_wait_for_draw (GtkWidget *widget);
```

Enters the main loop and waits for `widget` to be “drawn”. In this context that means it waits for the frame clock of `widget` to have run a full styling, layout and drawing cycle.

This function is intended to be used for syncing with actions that depend on `widget` relayouting or on interaction with the display server.

Parameters

widget	the widget to wait for

Since: 3.10

Types and Values

Filesystem utilities

Filesystem utilities — Functions for working with GIO

Functions

GMountOperation *	gtk_mount_operation_new ()
gboolean	gtk_mount_operation_is_showing ()
void	gtk_mount_operation_set_parent ()
GtkWindow *	gtk_mount_operation_get_parent ()
void	gtk_mount_operation_set_screen ()
GdkScreen *	gtk_mount_operation_get_screen ()
gboolean	gtk_show_uri ()
gboolean	gtk_show_uri_on_window ()

Properties

gboolean	is-showing	Read
GtkWindow *	parent	Read / Write
GdkScreen *	screen	Read / Write

Types and Values

struct	GtkMountOperation
struct	GtkMountOperationClass

Object Hierarchy

```
GObject
└── GMountOperation
    └── GtkMountOperation
```

Includes

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

Description

The functions and objects described here make working with GTK+ and GIO more convenient.

GtkMountOperation is needed when mounting volumes: It is an implementation of GMountOperation that can be used with GIO functions for mounting volumes such as `g_file_mount_enclosing_volume()`, `g_file_mount_mountable()`, `g_volume_mount()`, `g_mount_unmount_with_operation()` and others.

When necessary, GtkMountOperation shows dialogs to ask for passwords, questions or show processes blocking unmount.

gtk_show_uri_on_window() is a convenient way to launch applications for URIs.

Another object that is worth mentioning in this context is GdkAppLaunchContext, which provides visual feedback when lauching applications.

Functions

gtk_mount_operation_new ()

```
GMountOperation *
gtk_mount_operation_new (GtkWindow *parent);
```

Creates a new GtkMountOperation

Parameters

parent	transient parent of the window, or `NULL`.	[allow-none]

Returns

a new GtkMountOperation

Since: 2.14

---

gtk_mount_operation_is_showing ()

```
gboolean
gtk_mount_operation_is_showing (GtkMountOperation *op);
```

Returns whether the GtkMountOperation is currently displaying a window.

Parameters

op	a GtkMountOperation	

Returns

`TRUE` if `op` is currently displaying a window

Since: 2.14

---

gtk_mount_operation_set_parent ()

```
void
gtk_mount_operation_set_parent (GtkMountOperation *op,
                                GtkWindow *parent);
```

Sets the transient parent for windows shown by the GtkMountOperation.

Parameters

op	a GtkMountOperation	
parent	transient parent of the window, or `NULL`.	[allow-none]

Since: 2.14

---

gtk_mount_operation_get_parent ()

```
GtkWindow *
gtk_mount_operation_get_parent (GtkMountOperation *op);
```

Gets the transient parent used by the GtkMountOperation

Parameters

op	a GtkMountOperation

Returns

the transient parent for windows shown by `op` .

[transfer none]

Since: 2.14

---

gtk_mount_operation_set_screen ()

```
void
gtk_mount_operation_set_screen (GtkMountOperation *op,
                                GdkScreen *screen);
```

Sets the screen to show windows of the GtkMountOperation on.

Parameters

op	a GtkMountOperation
screen	a GdkScreen

Since: 2.14

---

gtk_mount_operation_get_screen ()

```
GdkScreen *
gtk_mount_operation_get_screen (GtkMountOperation *op);
```

Gets the screen on which windows of the GtkMountOperation will be shown.

Parameters

op	a GtkMountOperation

Returns

the screen on which windows of `op` are shown.

[transfer none]

Since: 2.14

---

gtk_show_uri ()

```
gboolean
gtk_show_uri (GdkScreen *screen,
              const gchar *uri,
              guint32 timestamp,
              GError **error);
```

`gtk_show_uri` is deprecated and should not be used in newly-written code.

A convenience function for launching the default application to show the uri. Like `gtk_show_uri_on_window()`, but takes a screen as transient parent instead of a window.

Note that this function is deprecated as it does not pass the necessary information for helpers to parent their dialog properly, when run from sandboxed applications for example.

Parameters

screen	screen to show the uri on or `NULL` for the default screen.	[allow-none]
uri	the uri to show	
timestamp	a timestamp to prevent focus stealing	
error	a GError that is returned in case of errors	

Returns

`TRUE` on success, `FALSE` on error

Since: 2.14

---

gtk_show_uri_on_window ()

```
gboolean
gtk_show_uri_on_window (GtkWindow *parent,
                        const char *uri,
                        guint32 timestamp,
                        GError **error);
```

This is a convenience function for launching the default application to show the uri. The uri must be of a form understood by GIO (i.e. you need to install gvfs to get support for uri schemes such as http:// or ftp://, as only local files are handled by GIO itself). Typical examples are

- `file:///home/gnome/pict.jpg`
- `http://www.gnome.org`
- `mailto:me@gnome.org`

Ideally the timestamp is taken from the event triggering the `gtk_show_uri()` call. If timestamp is not known you can take `GDK_CURRENT_TIME`.

This is the recommended call to be used as it passes information necessary for sandbox helpers to parent their dialogs properly.

Parameters

parent	parent window.	[allow-none]
uri	the uri to show	
timestamp	a timestamp to prevent focus stealing	
error	a GError that is returned in case of errors	

Returns

`TRUE` on success, `FALSE` on error

Since: 3.22

Types and Values

struct GtkMountOperation

```
struct GtkMountOperation;
```

This should not be accessed directly. Use the accessor functions below.

---

struct GtkMountOperationClass

```
struct GtkMountOperationClass {
  GMountOperationClass parent_class;
};
```

Members

Property Details

The `“is-showing”` property

```
  “is-showing”               gboolean
```

Are we showing a dialog.

Flags: Read

Default value: FALSE

The `“parent”` property

```
  “parent”                   GtkWindow *
```

The parent window.

Flags: Read / Write

The “screen” property

“screen” GdkScreen *

The screen where this window will be displayed.

Flags: Read / Write
