GtkIMContext

GtkIMContext — Base class for input method contexts

Functions

void	gtk_im_context_set_client_window ()
void	gtk_im_context_get_preedit_string ()
gboolean	gtk_im_context_filter_keypress ()
void	gtk_im_context_focus_in ()
void	gtk_im_context_focus_out ()
void	gtk_im_context_reset ()
void	gtk_im_context_set_cursor_location ()
void	gtk_im_context_set_use_preedit ()
void	gtk_im_context_set_surrounding ()
gboolean	gtk_im_context_get_surrounding ()
gboolean	gtk_im_context_delete_surrounding ()

Properties

GtkInputHints	input-hints	Read / Write
GtkInputPurpose	input-purpose	Read / Write

Signals

void	commit	Run Last
gboolean	delete-surrounding	Run Last
void	preedit-changed	Run Last
void	preedit-end	Run Last
void	preedit-start	Run Last
gboolean	retrieve-surrounding	Run Last

Types and Values

struct	GtkIMContext
struct	GtkIMContextClass
struct	GtkIMContextInfo

Object Hierarchy

```
GObject
    GtkIMContext
        GtkIMContextSimple
        GtkIMMulticontext
```

Includes

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

Description

GtkIMContext defines the interface for GTK+ input methods. An input method is used by GTK+ text input widgets like GtkEntry to map from key events to Unicode character strings.

The default input method can be set programmatically via the "gtk-im-module" GtkSettings property. Alternatively, you may set the GTK_IM_MODULE environment variable as documented in Running GTK+ Applications.

The GtkEntry "im-module" and GtkTextView "im-module" properties may also be used to set input methods for specific widget instances. For instance, a certain entry widget might be expected to contain certain characters which would be easier to input with a certain input method.

An input method may consume multiple key events in sequence and finally output the composed result. This is called preediting, and an input method may provide feedback about this process by displaying the intermediate composition states as preedit text. For instance, the default GTK+ input method implements the input of arbitrary Unicode code points by holding down the Control and Shift keys and then typing "U" followed by the hexadecimal digits of the code point. When releasing the Control and Shift keys, preediting ends and the character is inserted as text. Ctrl+Shift+u20AC for example results in the € sign.

Additional input methods can be made available for use by GTK+ widgets as loadable modules. An input method module is a small shared library which implements a subclass of GtkIMContext or GtkIMContextSimple and exports these four functions:

```
void im_module_init(GTypeModule *module);
```

This function should register the GType of the GtkIMContext subclass which implements the input method by means of `g_type_module_register_type()`. Note that `g_type_register_static()` cannot be used as the type needs to be registered dynamically.

```
void im_module_exit(void);
```

Here goes any cleanup code your input method might require on module unload.

```
void im_module_list(const GtkIMContextInfo ***contexts, int *n_contexts)
{
  *contexts = info_list;
  *n_contexts = G_N_ELEMENTS (info_list);
}
```

This function returns the list of input methods provided by the module. The example implementation above shows a common solution and simply returns a pointer to statically defined array of GtkIMContextInfo items for each provided input method.

```
GtkIMContext * im_module_create(const gchar *context_id);
```

This function should return a pointer to a newly created instance of the GtkIMContext subclass identified by `context_id` . The context ID is the same as specified in the GtkIMContextInfo array returned by `im_module_list()`.

After a new loadable input method module has been installed on the system, the configuration file `gtk.immodules` needs to be regenerated by gtk-query-immodules-3.0, in order for the new input method to become available to GTK+ applications.

Functions

gtk_im_context_set_client_window ()

```
void
gtk_im_context_set_client_window (GtkIMContext *context,
                                  GdkWindow *window);
```

Set the client window for the input context; this is the GdkWindow in which the input appears. This window is used in order to correctly position status windows, and may also be used for purposes internal to the input method.

Parameters

context	a GtkIMContext	
window	the client window. This may be `NULL` to indicate that the previous client window no longer exists.	[allow-none]

gtk_im_context_get_preedit_string ()

```
void
gtk_im_context_get_preedit_string (GtkIMContext *context,
                                   gchar **str,
                                   PangoAttrList **attrs,
                                   gint *cursor_pos);
```

Retrieve the current preedit string for the input context, and a list of attributes to apply to the string. This string should be displayed inserted at the insertion point.

Parameters

context	a GtkIMContext	
str	location to store the retrieved string. The string retrieved must be freed with `g_free()`.	[out][transfer full]
attrs	location to store the retrieved attribute list. When you are done with this list, you must unreference it with `pango_attr_list_unref()`.	[out][transfer full]
cursor_pos	location to store position of cursor (in characters) within the preedit string.	[out]

---

gtk_im_context_filter_keypress ()

```
gboolean
gtk_im_context_filter_keypress (GtkIMContext *context,
                                GdkEventKey *event);
```

Allow an input method to internally handle key press and release events. If this function returns `TRUE`, then no further processing should be done for this key event.

Parameters

context	a GtkIMContext
event	the key event

Returns

`TRUE` if the input method handled the key event.

---

gtk_im_context_focus_in ()

```
void
gtk_im_context_focus_in (GtkIMContext *context);
```

Notify the input method that the widget to which this input context corresponds has gained focus. The input method may, for example, change the displayed feedback to reflect this change.

Parameters

context	a GtkIMContext

---

gtk_im_context_focus_out ()

```
void
gtk_im_context_focus_out (GtkIMContext *context);
```

Notify the input method that the widget to which this input context corresponds has lost focus. The input method may, for example, change the displayed feedback or reset the contexts state to reflect this change.

Parameters

context	a GtkIMContext

gtk_im_context_reset ()

```
void
gtk_im_context_reset (GtkIMContext *context);
```

Notify the input method that a change such as a change in cursor position has been made. This will typically cause the input method to clear the preedit state.

Parameters

context	a GtkIMContext

gtk_im_context_set_cursor_location ()

```
void
gtk_im_context_set_cursor_location (GtkIMContext *context,
                                    const GdkRectangle *area);
```

Notify the input method that a change in cursor position has been made. The location is relative to the client window.

Parameters

context	a GtkIMContext
area	new location

gtk_im_context_set_use_preedit ()

```
void
gtk_im_context_set_use_preedit (GtkIMContext *context,
                                gboolean use_preedit);
```

Sets whether the IM context should use the preedit string to display feedback. If `use_preedit` is FALSE (default is TRUE), then the IM context may use some other method to display feedback, such as displaying it in a child of the root window.

Parameters

context	a GtkIMContext
use_preedit	whether the IM context should use the preedit string.

gtk_im_context_set_surrounding ()

```
void
gtk_im_context_set_surrounding (GtkIMContext *context,
                                const gchar *text,
                                gint len,
                                gint cursor_index);
```

Sets surrounding context around the insertion point and preedit string. This function is expected to be called in response to the GtkIMContext::retrieve_surrounding signal, and will likely have no effect if called at other times.

Parameters

context	a GtkIMContext
text	text surrounding the insertion point, as UTF-8. the preedit string should not be included within `text` .
len	the length of `text` , or -1 if `text` is nul-terminated

cursor_index	the byte index of the insertion cursor within `text` .

gtk_im_context_get_surrounding ()

```
gboolean
gtk_im_context_get_surrounding (GtkIMContext *context,
                                gchar **text,
                                gint *cursor_index);
```

Retrieves context around the insertion point. Input methods typically want context in order to constrain input text based on existing text; this is important for languages such as Thai where only some sequences of characters are allowed.

This function is implemented by emitting the GtkIMContext::retrieve_surrounding signal on the input method; in response to this signal, a widget should provide as much context as is available, up to an entire paragraph, by calling `gtk_im_context_set_surrounding()`. Note that there is no obligation for a widget to respond to the ::retrieve_surrounding signal, so input methods must be prepared to function without context.

Parameters

context	a GtkIMContext	
text	location to store a UTF-8 encoded string of text holding context around the insertion point. If the function returns `TRUE`, then you must free the result stored in this location with `g_free()`.	[out][transfer full]
cursor_index	location to store byte index of the insertion cursor within `text` .	[out]

Returns

`TRUE` if surrounding text was provided; in this case you must free the result stored in *text.

gtk_im_context_delete_surrounding ()

```
gboolean
gtk_im_context_delete_surrounding (GtkIMContext *context,
                                   gint offset,
                                   gint n_chars);
```

Asks the widget that the input context is attached to to delete characters around the cursor position by emitting the GtkIMContext::delete_surrounding signal. Note that `offset` and `n_chars` are in characters not in bytes which differs from the usage other places in GtkIMContext.

In order to use this function, you should first call `gtk_im_context_get_surrounding()` to get the current context, and call this function immediately afterwards to make sure that you know what you are deleting. You should also account for the fact that even if the signal was handled, the input context might not have deleted all the characters that were requested to be deleted.

This function is used by an input method that wants to make subsitutions in the existing text in response to new input. It is not useful for applications.

Parameters

context	a GtkIMContext
offset	offset from cursor position in chars; a negative value means start before the cursor.
n_chars	number of characters to delete.

Returns

`TRUE` if the signal was handled.

Types and Values

struct GtkIMContext

```
struct GtkIMContext;
```

struct GtkIMContextClass

```
struct GtkIMContextClass {
  /* Signals */
  void     (*preedit_start)        (GtkIMContext *context);
  void     (*preedit_end)          (GtkIMContext *context);
  void     (*preedit_changed)      (GtkIMContext *context);
  void     (*commit)               (GtkIMContext *context, const gchar *str);
  gboolean (*retrieve_surrounding) (GtkIMContext *context);
  gboolean (*delete_surrounding)   (GtkIMContext *context,
                                    gint          offset,
                                    gint          n_chars);

  /* Virtual functions */
  void     (*set_client_window)   (GtkIMContext   *context,
                                   GdkWindow      *window);
  void     (*get_preedit_string)  (GtkIMContext   *context,
                                   gchar         **str,
                                   PangoAttrList **attrs,
                                   gint           *cursor_pos);
  gboolean (*filter_keypress)     (GtkIMContext   *context,
                                   GdkEventKey    *event);
  void     (*focus_in)            (GtkIMContext   *context);
  void     (*focus_out)           (GtkIMContext   *context);
  void     (*reset)               (GtkIMContext   *context);
  void     (*set_cursor_location) (GtkIMContext   *context,
                                   GdkRectangle   *area);
  void     (*set_use_preedit)     (GtkIMContext   *context,
                                   gboolean        use_preedit);
  void     (*set_surrounding)     (GtkIMContext   *context,
                                   const gchar    *text,
                                   gint            len,
                                   gint            cursor_index);
  gboolean (*get_surrounding)     (GtkIMContext   *context,
                                   gchar         **text,
                                   gint           *cursor_index);
};
```

Members

`preedit_start` () — Default handler of the "preedit-start" signal.

`preedit_end` () — Default handler of the "preedit-end" signal.

`preedit_changed` () — Default handler of the "preedit-changed" signal.

`commit` () — Default handler of the "commit" signal.

`retrieve_surrounding` () — Default handler of the "retrieve-surrounding" signal.

`delete_surrounding` () — Default handler of the "delete-surrounding" signal.

`set_client_window` () — Called via `gtk_im_context_set_client_window()` when the input window where the entered text will appear changes. Override this to keep track of the current input window, for instance for the purpose of positioning a status display of your input method.

`get_preedit_string` () — Called via `gtk_im_context_get_preedit_string()` to retrieve the text currently being preedited for display at the cursor position. Any input method which composes complex characters or any other compositions from multiple sequential key presses should override this method to provide feedback.

`filter_keypress` () — Called via `gtk_im_context_filter_keypress()` on every key press or release event. Every non-trivial input method needs to override this in order to implement the mapping from key events to text. A return value of `TRUE` indicates to the caller that the event was consumed by the input method. In that case, the "commit" signal should be emitted upon completion of a key sequence to pass the resulting text back to the input widget. Alternatively, `FALSE` may be returned to indicate that the event wasn't handled by the input method. If a builtin mapping exists for the key, it is used to produce a character.

`focus_in` () — Called via `gtk_im_context_focus_in()` when the input widget has gained focus. May be overridden to keep track of the current focus.

`focus_out` () — Called via `gtk_im_context_focus_out()` when the input widget has lost focus. May be overridden to keep track of the current focus.

`reset` () — Called via `gtk_im_context_reset()` to signal a change such as a change in cursor position. An input method that implements preediting should override this method to clear the preedit state on reset.

`set_cursor_location` () — Called via `gtk_im_context_set_cursor_location()` to inform the input method of the current cursor location relative to the client window. May be overridden to implement the display of popup windows at the cursor position.

`set_use_preedit` () — Called via `gtk_im_context_set_use_preedit()` to control the use of the preedit string. Override this to display feedback by some other means if turned off.

`set_surrounding` () — Called via `gtk_im_context_set_surrounding()` in response to signal "retrieve-surrounding" to update the input method's idea of the context around the cursor. It is not necessary to override this method even with input methods which implement context-dependent behavior. The base implementation is sufficient for `gtk_im_context_get_surrounding()` to work.

`get_surrounding` () — Called via `gtk_im_context_get_surrounding()` to update the context around the cursor location. It is not necessary to override this method even with input methods which implement context-dependent behavior. The base implementation emits "retrieve-surrounding" and records the context received by the subsequent invocation of `get_surrounding` .

struct GtkIMContextInfo

```
struct GtkIMContextInfo {
  const gchar *context_id;
  const gchar *context_name;
  const gchar *domain;
  const gchar *domain_dirname;
  const gchar *default_locales;
};
```

Bookkeeping information about a loadable input method.

Members

const gchar *`context_id`;	The unique identification string of the input method.
const gchar *`context_name`;	The human-readable name of the input method.
const gchar *`domain`;	Translation domain to be used with `dgettext()`
const gchar *`domain_dirname`;	Name of locale directory for use with `bindtextdomain()`
const gchar *`default_locales`;	A colon-separated list of locales where this input method should be the default. The asterisk "*" sets the default for all locales.

Property Details

The "input-hints" property

```
  "input-hints"              GtkInputHints
```

Hints for the text field behaviour.

Flags: Read / Write

The "input-purpose" property

```
  "input-purpose"            GtkInputPurpose
```

Purpose of the text field.

Flags: Read / Write

Default value: GTK_INPUT_PURPOSE_FREE_FORM

Signal Details

The "commit" signal

```
void
user_function (GtkIMContext *context,
               gchar        *str,
               gpointer      user_data)
```

The ::commit signal is emitted when a complete input sequence has been entered by the user. This can be a single character immediately after a key press or the final result of preediting.

Parameters

context	the object on which the signal is emitted
str	the completed character(s) entered by the user
user_data	user data set when the signal handler was connected.

Flags: Run Last

The "delete-surrounding" signal

```
gboolean
user_function (GtkIMContext *context,
               gint          offset,
               gint          n_chars,
               gpointer      user_data)
```

The ::delete-surrounding signal is emitted when the input method needs to delete all or part of the context surrounding the cursor.

Parameters

context	the object on which the signal is emitted
offset	the character offset from the cursor position of the text to be deleted. A negative value indicates a position before the cursor.
n_chars	the number of characters to be deleted
user_data	user data set when the signal handler was connected.

Returns

`TRUE` if the signal was handled.

Flags: Run Last

The “preedit-changed” signal

```
void
user_function (GtkIMContext *context,
               gpointer      user_data)
```

The ::preedit-changed signal is emitted whenever the preedit sequence currently being entered has changed. It is also emitted at the end of a preedit sequence, in which case `gtk_im_context_get_preedit_string()` returns the empty string.

Parameters

context	the object on which the signal is emitted
user_data	user data set when the signal handler was connected.

Flags: Run Last

The “preedit-end” signal

```
void
user_function (GtkIMContext *context,
               gpointer      user_data)
```

The ::preedit-end signal is emitted when a preediting sequence has been completed or canceled.

Parameters

context	the object on which the signal is emitted
user_data	user data set when the signal handler was connected.

Flags: Run Last

The “preedit-start” signal

```
void
user_function (GtkIMContext *context,
               gpointer      user_data)
```

The ::preedit-start signal is emitted when a new preediting sequence starts.

Parameters

context	the object on which the signal is emitted
user_data	user data set when the signal handler was connected.

Flags: Run Last

The “retrieve-surrounding” signal

```
gboolean
user_function (GtkIMContext *context,
               gpointer      user_data)
```

The ::retrieve-surrounding signal is emitted when the input method requires the context surrounding the cursor. The callback should set the input method surrounding context by calling the `gtk_im_context_set_surrounding()` method.

Parameters

context	the object on which the signal is emitted
user_data	user data set when the signal handler was connected.

Returns

`TRUE` if the signal was handled.

Flags: Run Last
