# GTK::V3::Gdk::GdkEventTypes

## Event Structures — Data structures specific to each type of event

### Table of Contents



# Synopsis

```
my GTK::V3::Gtk::GtkWindow $top-window .= new(:empty);
$top-window.set-title('Hello GTK!');
# ... etcetera ...

# Register a signal handler for a window event
$top-window.register-signal( self, 'handle-keypress', 'key-press-event');

method handle-keypress ( :$widget, GdkEvent :$event ) {
  if $event.event-any.type ~~ GDK_KEY_PRESS and
     $event.event-key.keyval eq 's' {

    # key 's' pressed, stop process ...
  }
}
```

The handler signature can also be defined as

```
method handle-keypress ( :$widget, GdkEventKey :$event ) {
  if $event.type ~~ GDK_KEY_PRESS and $event.keyval eq 's' {

    # key 's' pressed, stop process ...
  }
}
```

# class GTK::V3::Gdk::GdkEventTypes

# Enums, Structs and Unions

## Enum GdkEventType Specifies the type of the event.

Do not confuse these events with the signals that GTK+ widgets emit. Although many of these events result in corresponding signals being emitted, the events are often transformed or filtered along the way.

In some language bindings, the values GDK_2BUTTON_PRESS and GDK_3BUTTON_PRESS would translate into something syntactically invalid (eg Gdk.EventType.2ButtonPress, where a symbol is not allowed to start with a number). In that case, the aliases GDK_DOUBLE_BUTTON_PRESS and GDK_TRIPLE_BUTTON_PRESS can be used instead.

- GDK_NOTHING; a special code to indicate a null event.
- GDK_DELETE; the window manager has requested that the toplevel window be hidden or destroyed, usually when the user clicks on a special icon in the title bar.
- GDK_DESTROY; the window has been destroyed.
- GDK_EXPOSE; all or part of the window has become visible and needs to be redrawn.
- GDK_MOTION_NOTIFY; the pointer (usually a mouse) has moved.
- GDK_BUTTON_PRESS; a mouse button has been pressed.
- GDK_2BUTTON_PRESS; a mouse button has been double-clicked (clicked twice within a short period of time). Note that each click also generates a GDK_BUTTON_PRESS event.
- GDK_DOUBLE_BUTTON_PRESS; alias for GDK_2BUTTON_PRESS, added in 3.6.
- GDK_3BUTTON_PRESS; a mouse button has been clicked 3 times in a short

period of time. Note that each click also generates a GDK_BUTTON_PRESS event.

- GDK_TRIPLE_BUTTON_PRESS; alias for GDK_3BUTTON_PRESS, added in 3.6.
- GDK_BUTTON_RELEASE; a mouse button has been released.
- GDK_KEY_PRESS; a key has been pressed.
- GDK_KEY_RELEASE; a key has been released.
- GDK_ENTER_NOTIFY; the pointer has entered the window.
- GDK_LEAVE_NOTIFY; the pointer has left the window.
- GDK_FOCUS_CHANGE; the keyboard focus has entered or left the window.
- GDK_CONFIGURE; the size, position or stacking order of the window has changed. Note that GTK+ discards these events for GDK_WINDOW_CHILD windows.
- GDK_MAP; the window has been mapped.
- GDK_UNMAP; the window has been unmapped.
- GDK_PROPERTY_NOTIFY; a property on the window has been changed or deleted.
- GDK_SELECTION_CLEAR; the application has lost ownership of a selection.
- GDK_SELECTION_REQUEST; another application has requested a selection.
- GDK_SELECTION_NOTIFY; a selection has been received.
- GDK_PROXIMITY_IN; an input device has moved into contact with a sensing surface (e.g. a touchscreen or graphics tablet).
- GDK_PROXIMITY_OUT; an input device has moved out of contact with a sensing surface.
- GDK_DRAG_ENTER; the mouse has entered the window while a drag is in progress.
- GDK_DRAG_LEAVE; the mouse has left the window while a drag is in progress.

- GDK_DRAG_MOTION; the mouse has moved in the window while a drag is in progress.
- GDK_DRAG_STATUS; the status of the drag operation initiated by the window has changed.
- GDK_DROP_START; a drop operation onto the window has started.
- GDK_DROP_FINISHED; the drop operation initiated by the window has completed.
- GDK_CLIENT_EVENT; a message has been received from another application.
- GDK_VISIBILITY_NOTIFY; the window visibility status has changed.
- GDK_SCROLL; the scroll wheel was turned.
- GDK_WINDOW_STATE; the state of a window has changed. See GdkWindowState for the possible window states.
- GDK_SETTING. a setting has been modified.
- GDK_OWNER_CHANGE; the owner of a selection has changed. This event type was added in 2.6
- GDK_GRAB_BROKEN; a pointer or keyboard grab was broken. This event type was added in 2.8.
- GDK_DAMAGE; the content of the window has been changed. This event type was added in 2.14.
- GDK_TOUCH_BEGIN; A new touch event sequence has just started. This event type was added in 3.4.
- GDK_TOUCH_UPDATE; A touch event sequence has been updated. This event type was added in 3.4.
- GDK_TOUCH_END; A touch event sequence has finished. This event type was added in 3.4.
- GDK_TOUCH_CANCEL; A touch event sequence has been canceled. This event type was added in 3.4.
- GDK_TOUCHPAD_SWIPE; A touchpad swipe gesture event, the current state is determined by its phase field. This event type was added in 3.18.
- GDK_TOUCHPAD_PINCH; A touchpad pinch gesture event, the current

state is determined by its phase field. This event type was added in 3.18.

- GDK_PAD_BUTTON_PRESS; A tablet pad button press event. This event type was added in 3.22.
- GDK_PAD_BUTTON_RELEASE; A tablet pad button release event. This event type was added in 3.22.
- GDK_PAD_RING; A tablet pad axis event from a "ring". This event type was added in 3.22.
- GDK_PAD_STRIP; A tablet pad axis event from a "strip". This event type was added in 3.22.
- GDK_PAD_GROUP_MODE; A tablet pad group mode change. This event type was added in 3.22.
- GDK_EVENT_LAST; Marks the end of the GdkEventType enumeration. Added in 2.18

## class GdkEventAny

Contains the fields which are common to all event classes. This comes in handy to check its type for instance.

- GdkEventType $.type; the type of the event.
- N-GObject $.window; the window which received the event.
- Int $.send_event; TRUE if the event was sent explicitly.

## class GdkEventKey

Describes a key press or key release event. The type of the event will be one of GDK_KEY_PRESS or GDK_KEY_RELEASE.

- GdkEventType $.type
- N-GObject $.window
- Int $.send_event
- UInt $.time; the time of the event in milliseconds.
- UInt $.state; a bit-mask representing the state of the modifier keys (e.g. Control, Shift and Alt) and the pointer buttons. See GdkModifierType. [type GdkModifierType].

- UInt $.keyval; the key that was pressed or released. See the gdk/gdkkeysyms.h header file for a complete list of GDK key codes.
- Int $.length; the length of string.
- Str $.string; deprecated.
- UInt $.hardware_keycode; the raw code of the key that was pressed or released.
- UInt $.group; the keyboard group.
- UInt $.is_modifier; a flag that indicates if hardware_keycode is mapped to a modifier. Since 2.10

## class GdkEventButton

Used for mouse button press and button release events. The type will be one of GDK_BUTTON_PRESS, GDK_2BUTTON_PRESS, GDK_3BUTTON_PRESS or GDK_BUTTON_RELEASE,

Double and triple-clicks result in a sequence of events being received. For double-clicks the order of events will be: GDK_BUTTON_PRESS, GDK_BUTTON_RELEASE, GDK_BUTTON_PRESS, GDK_2BUTTON_PRESS and GDK_BUTTON_RELEASE.

Note that the first click is received just like a normal button press, while the second click results in a GDK_2BUTTON_PRESS being received just after the GDK_BUTTON_PRESS.

Triple-clicks are very similar to double-clicks, except that GDK_3BUTTON_PRESS is inserted after the third click. The order of the events is: GDK_BUTTON_PRESS, GDK_BUTTON_RELEASE, GDK_BUTTON_PRESS, GDK_2BUTTON_PRESS, GDK_BUTTON_RELEASE, GDK_BUTTON_PRESS, GDK_3BUTTON_PRESS and GDK_BUTTON_RELEASE.

For a double click to occur, the second button press must occur within 1/4 of a second of the first. For a triple click to occur, the third button press must also occur within 1/2 second of the first button press.

To handle e.g. a triple mouse button presses, all events can be ignored except GDK_3BUTTON_PRESS

```
method handle-keypress ( :$widget, GdkEventButton :$event ) {
  # check if left mouse button was pressed three times
  if $event.type ~~ GDK_3BUTTON_PRESS and $event.button == 1 {
    ...
  }
}
```

- GdkEventType $.type;
- N-GObject $.window;
- Int $.send_event;
- UInt $.time; the time of the event in milliseconds.
- Num $.x; the x coordinate of the pointer relative to the window.
- Num $.y; the y coordinate of the pointer relative to the window.
- Pointer[Num] $.axes; x , y translated to the axes of device , or NULL if device is the mouse.
- UInt $.state; a bit-mask representing the state of the modifier keys (e.g. Control, Shift and Alt) and the pointer buttons. See GdkModifierType.
- UInt $.button; the button which was pressed or released, numbered from 1 to 5. Normally button 1 is the left mouse button, 2 is the middle button, and 3 is the right button. On 2-button mice, the middle button can often be simulated by pressing both mouse buttons together.
- N-GObject $.device; the master device that the event originated from. Use gdk_event_get_source_device() to get the slave device.
- Num $.x_root; the x coordinate of the pointer relative to the root of the screen.
- Num $.y_root; the y coordinate of the pointer relative to the root of the screen.

## class GdkEventTouch Used for touch events. type field will be one of GDK_TOUCH_BEGIN, GDK_TOUCH_UPDATE, GDK_TOUCH_END or GDK_TOUCH_CANCEL.

Touch events are grouped into sequences by means of the sequence field, which can also be obtained with gdk_event_get_event_sequence(). Each sequence begins with a GDK_TOUCH_BEGIN event, followed by any number of

GDK_TOUCH_UPDATE events, and ends with a GDK_TOUCH_END (or GDK_TOUCH_CANCEL) event. With multitouch devices, there may be several active sequences at the same time.

- GdkEventType $.type; the type of the event (GDK_TOUCH_BEGIN, GDK_TOUCH_UPDATE, GDK_TOUCH_END, GDK_TOUCH_CANCEL)
- N-GObject $.window;
- Int $.send_event;
- uint32 $.time; the time of the event in milliseconds.
- num64 $.x; the x coordinate of the pointer relative to the window
- num64 $.y; the y coordinate of the pointer relative to the window
- Pointer[num64] $.axes; x , y translated to the axes of device , or NULL if device is the mouse
- uint32 state; a bit-mask representing the state of the modifier keys (e.g. Control, Shift and Alt) and the pointer buttons. See GdkModifierType.
- Pointer $.sequence; the event sequence that the event belongs to
- int32 emulating_pointer; whether the event should be used for emulating pointer event (0 or 1)
- N-GObject $.device; the master device that the event originated from. Use gdk_event_get_source_device() to get the slave device.
- num64 $.x_root; the x coordinate of the pointer relative to the root of the screen
- num64 $.y_root; the y coordinate of the pointer relative to the root of the screen

## GdkEvent

The event structures contain data specific to each type of event in GDK. The type is a union of all structures explained above.