**XGrabKey, XUngrabKey – grab keyboard keys**

**XGrabKey(***display***,** *keycode***,** *modifiers***,** *grab_window***,** *owner_events***,** *pointer_mode***,**
*keyboard_mode***)**
**Display ****display***;**
**int** *keycode***;**
**unsigned int** *modifiers***;**
**Window** *grab_window***;**
**Bool** *owner_events***;**
**int** *pointer_mode***,** *keyboard_mode***;**

XUngrabKey(*display*, *keycode*, *modifiers*, *grab_window*)
Display **display*;
int *keycode*;
unsigned int *modifiers*;
Window *grab_window*;

| | |
|---|---|
| *display* | Specifies the connection to the X server. |
| *grab_window* | Specifies the grab window. |
| *keyboard_mode* | Specifies further processing of keyboard events. You can pass **GrabModeSync** or **GrabModeAsync**. |
| *keycode* | Specifies the KeyCode or **AnyKey**. |
| *modifiers* | Specifies the set of keymasks or **AnyModifier**. The mask is the bitwise inclusive OR of the valid keymask bits. |
| *owner_events* | Specifies a Boolean value that indicates whether the keyboard events are to be reported as usual. |
| *pointer_mode* | Specifies further processing of pointer events. You can pass **GrabModeSync** or **GrabModeAsync**. |

**The XGrabKey** function establishes a passive grab on the keyboard. In the future, the keyboard is actively grabbed (as for **XGrabKeyboard**), the last-keyboard-grab time is set to the time at which the key was pressed (as transmitted in the **KeyPress** event), and the **KeyPress** event is reported if all of the following conditions are true:

- The keyboard is not grabbed and the specified key (which can itself be a modifier key) is logically pressed when the specified modifier keys are logically down, and no other modifier keys are logically down.
- Either the grab_window is an ancestor of (or is) the focus window, or the grab_window is a descendant of the focus window and contains the pointer.
- A passive grab on the same key combination does not exist on any ancestor of grab_window.

The interpretation of the remaining arguments is as for **XGrabKeyboard**. The active grab is terminated automatically when the logical state of the keyboard has the specified key released (independent of the logical state of the modifier keys).

Note that the logical state of a device (as seen by client applications) may lag the physical state if device event processing is frozen.

A modifiers argument of **AnyModifier** is equivalent to issuing the request for all possible modifier combinations (including the combination of no modifiers). It is not required that all modifiers specified have currently assigned KeyCodes. A keycode argument of **AnyKey** is equivalent to issuing the request for all possible KeyCodes. Otherwise, the specified keycode must be in the range specified by min_keycode and max_keycode in the connection setup, or a **BadValue** error results.

If some other client has issued a **XGrabKey** with the same key combination on the same window, a **BadAccess** error results. When using **AnyModifier** or **AnyKey**, the request fails completely, and a

**BadAccess** error results (no grabs are established) if there is a conflicting grab for any combination.

**XGrabKey** can generate **BadAccess**, **BadValue**, and **BadWindow** errors.

The **XUngrabKey** function releases the key combination on the specified window if it was grabbed by this client. It has no effect on an active grab. A modifiers of **AnyModifier** is equivalent to issuing the request for all possible modifier combinations (including the combination of no modifiers). A keycode argument of **AnyKey** is equivalent to issuing the request for all possible key codes.

**XUngrabKey** can generate **BadValue** and **BadWindow** error.

**BadAccess** A client attempted to grab a key/button combination already grabbed by another client. **BadValue** Some numeric value falls outside the range of values accepted by the request. Unless a specific range is specified for an argument, the full range defined by the argument's type is accepted. Any argument defined as a set of alternatives can generate this error. **BadWindow** A value for a Window argument does not name a defined Window.

**XAllowAccess(3X11), XGrabButton(3X11), XGrabKeyboard(3X11), XGrabPointer(3X11)**
*Xlib – C Language X Interface*