NAME

XGrabButton, XUngrabButton - grab pointer buttons

SYNTAX

int XGrabButton(Display **display*, unsigned int *button*, unsigned int *modifiers*, Window *grab_window*, Bool *owner_events*, unsigned int *event_mask*, int *pointer_mode*, *keyboard_mode*, Window *confine_to*, Cursor *cursor*);

int XUngrabButton(Display **display*, unsigned int *button*, unsigned int *modifiers*, Window *grab_window*);

ARGUMENTS

button Specifies the pointer button that is to be grabbed or released or **AnyButton**.

confine_to Specifies the window to confine the pointer in or **None**.

cursor Specifies the cursor that is to be displayed or **None**.

display Specifies the connection to the X server.

event_mask Specifies which pointer events are reported to the client. The mask is the bitwise inclusive OR of the valid pointer event mask bits.

grab_window Specifies the grab window.

keyboard_mode
Specifies further processing of keyboard events. You can pass **GrabModeSync** or **GrabModeAsync**.

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 pointer events are to be reported as usual or reported with respect to the grab window if selected by the event mask.

pointer_mode Specifies further processing of pointer events. You can pass **GrabModeSync** or **GrabModeAsync**.

DESCRIPTION

The **XGrabButton** function establishes a passive grab. In the future, the pointer is actively grabbed (as

for **XGrabPointer**), the last-pointer-grab time is set to the time at which the button was pressed (as transmitted in the **ButtonPress** event), and the **ButtonPress** event is reported if all of the following conditions are true:

- The pointer is not grabbed, and the specified button is logically pressed when the specified modifier keys are logically down, and no other buttons or modifier keys are logically down.
- The grab_window contains the pointer.
- The confine_to window (if any) is viewable.
- A passive grab on the same button/key combination does not exist on any ancestor of grab_window.

The interpretation of the remaining arguments is as for **XGrabPointer**. The active grab is terminated automatically when the logical state of the pointer has all buttons released (independent of the state of the logical modifier keys), at which point a **ButtonRelease** event is reported to the grabbing window.

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

This request overrides all previous grabs by the same client on the same button/key combinations on the same window. A modifiers of **AnyModifier** is equivalent to issuing the grab 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 button of **AnyButton** is equivalent to issuing the request for all possible buttons. Otherwise, it is not required that the specified button currently be assigned to a physical button.

If some other client has already issued a **XGrabButton** with the same button/key combination on the same window, a **BadAccess** error results. When using **AnyModifier** or **AnyButton**, the request fails completely, and a **BadAccess** error results (no grabs are established) if there is a conflicting grab for any combination. **XGrabButton** has no effect on an active grab.

XGrabButton can generate **BadCursor**, **BadValue**, and **BadWindow** errors.

The **XUngrabButton** function releases the passive button/key combination on the specified window if it was grabbed by this client. A modifiers of **AnyModifier** is equivalent to issuing the ungrab request for all possible modifier combinations, including the combination of no modifiers. A button of **AnyButton** is equivalent to issuing the request for all possible buttons. **XUngrabButton** has no effect on an active grab.

XUngrabButton can generate **BadValue** and **BadWindow** errors.

DIAGNOSTICS

BadCursor A value for a Cursor argument does not name a defined Cursor.

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.

SEE ALSO

XAllowEvents(3), XGrabPointer(3), XGrabKey(3), XGrabKeyboard(3),
Xlib - C Language X Interface