XGrabDeviceKey, XUngrabDeviceKey – grab/ungrab extension input device Keys

XGrabDeviceKey(*display***,** *device***,** *Key***,** *modifiers***,** *modifier_device***,** *grab_window***,** *owner_events***,** *event_count***,** *event_list***,** *this_device_mode***,** *other_devices_mode***)**
Display ***display;**
XDevice ***device;**
unsigned int *Key***;**
unsigned int *modifiers***;**
XDevice ***modifier_device;**
Window *grab_window***;**
Bool *owner_events***;**
unsigned int *event_count***;**
XEventClass *event_list***;**
int *this_device_mode***,** *other_devices_mode***;**

XUngrabDeviceKey(*display***,** *device***,** *Key***,** *modifiers***,** *modifier_device***,** *grab_window***)**
Display ***display;**
XDevice ***device;**
unsigned int *Key***;**
unsigned int *modifiers***;**
XDevice ***modifier_device;**
Window *grab_window***;**

display **Specifies the connection to the X server.** *device* **Specifies the device that is to be grabbed or released** *Key* **Specifies the device Key that is to be grabbed or released or** *AnyKey***.** *modifiers* **Specifies the set of keymasks or** *AnyModifier***. The mask is the bitwise inclusive OR of the valid keymask bits. Valid bits are: ShiftMask, LockMask, ControlMask, Mod1Mask, Mod2Mask, Mod3Mask, Mod4Mask, Mod5Mask.** *modifier_device* **Specifies the device whose modifiers are to be used. If a modifier_device of NULL is specified, the X keyboard will be used as the modifier_device.** *grab_window* **Specifies the grab window.** *owner_events* **Specifies a Boolean value that indicates whether the device events are to be reported as usual or reported with respect to the grab window if selected by the event list.** *event_count* **Specifies the number of event classes in the event list.** *event_list* **Specifies which device events are reported to the client.** *this_device_mode* **Specifies further processing of events from this device. You can pass** *GrabModeSync* **or** *GrabModeAsync***.** *other_devices_mode* **Specifies further processing of events from other devices. You can pass** *GrabModeSync* **or** *GrabModeAsync***.**

The *XGrabDeviceKey* **request establishes a passive grab. In the future, the device is actively grabbed (as for** *XGrabDevice***, the last-device-grab time is set to the time at which the Key was pressed (as transmitted in the** *DeviceKeyPress* **event), and the** *DeviceKeyPress* **event is reported if all of the following conditions are true:**

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

The interpretation of the remaining arguments is as for *XGrabDevice* . The active grab is terminated automatically when the logical state of the device has the specified key released.

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

If the key is not *AnyKey*, it must be in the range specified by min_keycode and max_keycode as returned by the *XListInputDevices* request. Otherwise, a *BadValue* error results.

This request overrides all previous grabs by the same client on the same Key/modifier combinations on the same window. A modifier 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 key of *AnyKey* is equivalent to issuing the request for all possible keys. Otherwise, it is not required that the specified key currently be assigned to a physical Key.

If a modifier_device of NULL is specified, the X keyboard will be used as the modifier_device.

If some other client has already issued a *XGrabDeviceKey* with the same Key/modifier 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. *XGrabDeviceKey* has no effect on an active grab.

XGrabDeviceKey can generate *BadAccess, BadClass, BadDevice, BadMatch, BadValue*, and *BadWindow* errors. It returns *Success* on successful completion of the request. The *XUngrabDeviceKey* request releases the passive grab for a key/modifier combination on the specified window if it was grabbed by this client. A modifier of *AnyModifier* is equivalent to issuing the ungrab request for all possible modifier combinations, including the combination of no modifiers. A Key of *AnyKey* is equivalent to issuing the request for all possible Keys. *XUngrabDeviceKey* has no effect on an active grab.

If a modifier_device of NULL is specified, the X keyboard will be used as the modifier_device.

XUngrabDeviceKey can generate *BadDevice*, *BadMatch*, *BadValue* and *BadWindow* errors.

BadDevice **An invalid device was specified. The specified device does not exist or has not been opened by this client via** *XOpenInputDevice***. This error may also occur if the specified device is the X keyboard or X pointer device.** *BadMatch* **This error may occur if an** *XGrabDeviceKey* **request was made specifying a device that has no keys, or a modifier device that has no keys.** *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.**

XAllowDeviceEvents(3X),
XGrabDevice(3X),
XGrabDeviceButton(3X),
Programming with Xlib