

NAME

XkbSetIgnoreLockMods - Sets the modifiers that, if locked, are not to be reported in matching events to passive grabs

SYNOPSIS

Bool XkbSetIgnoreLockMods (Display **display*, unsigned int *device_spec*, unsigned int *affect_real*, unsigned int *real_values*, unsigned int *affect_virtual*, unsigned int *virtual_values*);

ARGUMENTS

display

connection to the X server

device_spec

device ID, or XkbUseCoreKbd

affect_real

mask of real modifiers affected by this call

real_values

values for affected real modifiers (1=>set, 0=>unset)

affect_virtual

mask of virtual modifiers affected by this call

virtual_values

values for affected virtual modifiers (1=>set, 0=>unset)

DESCRIPTION

The core protocol does not provide a way to exclude specific modifiers from grab calculations, with the result that locking modifiers sometimes have unanticipated side effects.

The IgnoreLockMods control specifies modifiers that should be excluded from grab calculations. These modifiers are also not reported in any core events except KeyPress and KeyRelease events that do not activate a passive grab and that do not occur while a grab is active.

Manipulate the IgnoreLockMods control via the *ignore_lock* field in the XkbControlsRec structure, then use *XkbSetControls* and *XkbGetControls* to query and change this control. Alternatively, use *XkbSetIgnoreLockMods*.

XkbSetIgnoreLockMods sends a request to the server to change the server's IgnoreLockMods control.

affect_real and *real_values* are masks of real modifier bits indicating which real modifiers are to be added and removed from the server's IgnoreLockMods control. Modifiers selected by both *affect_real* and *real_values* are added to the server's IgnoreLockMods control; those selected by *affect_real* but not by *real_values* are removed from the server's IgnoreLockMods control. Valid values for *affect_real* and *real_values* consist of any combination of the eight core modifier bits: ShiftMask, LockMask, ControlMask, Mod1Mask - Mod5Mask. *affect_virtual* and *virtual_values* are masks of virtual modifier bits indicating which virtual modifiers are to be added and removed from the server's IgnoreLockMods control. Modifiers selected by both *affect_virtual* and *virtual_values* are added to the server's IgnoreLockMods control; those selected by *affect_virtual* but not by *virtual_values* are removed from the server's IgnoreLockMods control. See below for a discussion of virtual modifier masks to use in *affect_virtual* and *virtual_values*. *XkbSetIgnoreLockMods* does not wait for a reply from the server. It returns True if the request was sent, and False otherwise.

Virtual modifiers are named by converting their string name to an X Atom and storing the Atom in the *names.vmods* array in an XkbDescRec structure. The position of a name Atom in the *names.vmods* array defines the bit position used to represent the virtual modifier and also the index used when accessing virtual modifier information in arrays: the name in the *i*-th (0 relative) entry of *names.vmods* is the *i*-th virtual modifier, represented by the mask (1<<*i*). Throughout Xkb, various functions have a parameter that is a mask representing virtual modifier choices. In each case, the *i*-th bit (0 relative) of the mask represents the *i*-th virtual modifier.

To set the name of a virtual modifier, use *XkbSetNames*, using XkbVirtualModNamesMask in *which* and the name in the *xkb* argument; to retrieve indicator names, use *XkbGetNames*.

STRUCTURES

The complete description of an Xkb keyboard is given by an XkbDescRec. The component structures in the XkbDescRec represent the major Xkb components.

```
typedef struct {
    struct _XDisplay * display; /* connection to X server */
    unsigned short flags; /* private to Xkb, do not modify */
    unsigned short device_spec; /* device of interest */
    KeyCode min_key_code; /* minimum keycode for device */
    KeyCode max_key_code; /* maximum keycode for device */
    XkbControlsPtr ctrls; /* controls */
    XkbServerMapPtr server; /* server keymap */
    XkbClientMapPtr map; /* client keymap */
    XkbIndicatorPtr indicators; /* indicator map */
    XkbNamesPtr names; /* names for all components */
    XkbCompatMapPtr compat; /* compatibility map */
}
```

```

    XkbGeometryPtr  geom;    /* physical geometry of keyboard */
} XkbDescRec, *XkbDescPtr;

```

The display field points to an X display structure. The flags field is private to the library: modifying flags may yield unpredictable results. The device_spec field specifies the device identifier of the keyboard input device, or `XkbUseCoreKeyboard`, which specifies the core keyboard device. The `min_key_code` and `max_key_code` fields specify the least and greatest keycode that can be returned by the keyboard.

Each structure component has a corresponding mask bit that is used in function calls to indicate that the structure should be manipulated in some manner, such as allocating it or freeing it. These masks and their relationships to the fields in the `XkbDescRec` are shown in Table 1.

Table 1 Mask Bits for
`XkbDescRec`

Mask Bit	XkbDescRec Field	Value
<code>XkbControlsMask</code>	<code>ctrls</code>	$(1L \ll 0)$
<code>XkbServerMapMask</code>	<code>server</code>	$(1L \ll 1)$
<code>XkbIClientMapMask</code>	<code>map</code>	$(1L \ll 2)$
<code>XkbIndicatorMapMask</code>	<code>indicators</code>	$(1L \ll 3)$
<code>XkbNamesMask</code>	<code>names</code>	$(1L \ll 4)$
<code>XkbCompatMapMask</code>	<code>compat</code>	$(1L \ll 5)$
<code>XkbGeometryMask</code>	<code>geom</code>	$(1L \ll 6)$
<code>XkbAllComponentsMask</code>	All Fields	$(0x7f)$

SEE ALSO

`XkbGetControls(3)`, `XkbGetNames(3)`, `XkbSetControls(3)`, `XkbSetNames(3)`