NAME

XkbNoteControlsChanges - Notes the changes in a changes structure when a client receives an XkbControlsNotify event

SYNOPSIS

void XkbNoteControlsChanges (XkbControlsChangesPtr *changes***, XkbControlsNotifyEvent ***_new_**,**
unsigned int *wanted***);**

ARGUMENTS

changes

records changes indicated by new

new

tells which things have changed

wanted

tells which parts of new to record in changes

DESCRIPTION

Whenever a field in the controls structure changes in the server's keyboard description, the server sends an XkbControlsNotify event to all interested clients.To receive XkbControlsNotify events under all possible conditions, use *XkbSelectEvents* and pass XkbControlsNotifyMask in both *bits_to_change* and *values_for_bits.*

To receive XkbControlsNotify events only under certain conditions, use *XkbSelectEventDetails* using XkbControlsNotify as the *event_type* and specifying the desired state changes in *bits_to_change* and *values_for_bits* using mask bits from Table 1.

Table 1 shows the actual values for the individual mask bits used to select controls for modification and to enable and disable the control. Note that the same mask bit is used to specify general modifications to the parameters used to configure the control (which), and to enable and disable the control (enabled_ctrls). The anomalies in the table (no "ok" in column) are for controls that have no configurable attributes; and for controls that are not boolean controls and therefore cannot be enabled or disabled.

Table 1 Controls Mask

Bits

Mask which enabledValue

Bit	changed_ctrls	ctrls	
XkbRepeatKeysMask	ok	ok	(1L<<0)
XkbSlowKeysMask	ok	ok	(1L<<1)
XkbBounceKeysMask	ok	ok	(1L<<2)
XkbStickyKeysMask	ok	ok	(1L<<3)
XkbMouseKeysMask	ok	ok	(1L<<4)
XkbMouseKeysAccelMask	ok	ok	(1L<<5)
XkbAccessXKeysMask	ok	ok	(1L<<6)
XkbAccessXTimeoutMask	ok	ok	(1L<<7)
XkbAccessXFeedbackMask	ok	ok	(1L<<8)
XkbAudibleBellMask		ok	(1L<<9)
XkbOverlay1Mask		ok	(1L<<10)
XkbOverlay2Mask		ok	(1L<<11)
XkbIgnoreGroupLockMask		ok	(1L<<12)
XkbGroupsWrapMask	ok		(1L<<27)
XkbInternalModsMask	ok		(1L<<28)
XkbIgnoreLockModsMask	ok		(1L<<29)
XkbPerKeyRepeatMask	ok		(1L<<30)
XkbControlsEnabledMask	ok		(1L<<31)
XkbAccessXOptionsMask	ok	ok	(XkbStickyKeysMask | XkbAccessXFeedbackMask)
XkbAllBooleanCtrlsMask		ok	(0x00001FFF)
XkbAllControlsMask	ok		(0xF8001FFF)

The *changed_ctrls* field specifies the controls components that have changed and consists of bits taken from the masks defined in Table 1 with "ok" in the *changed_ctrls* column.

The controls currently enabled in the server are reported in the *enabled_ctrls* field. If any controls were just enabled or disabled (that is, the contents of the *enabled_ctrls* field changed), they are flagged in the *enabled_ctrl_changes* field. The valid bits for these fields are the masks listed in Table 1 with "ok" in the *enabled_ctrls* column. The *num_groups* field reports the number of groups bound to the key belonging to the most number of groups and is automatically updated when the keyboard mapping changes.

If the change was caused by a request from a client, the *keycode* and *event_type* fields are set to zero and the *req_major* and *req_minor* fields identify the request. The *req_major value is the same as the major extension opcode. Otherwise, event_type* is set to the type of event that caused the change (one of KeyPress, KeyRelease, DeviceKeyPress, DeviceKeyRelease, ButtonPress or ButtonRelease), and

req_major and *req_minor* are undefined. If *event_type* is KeyPress, KeyRelease, DeviceKeyPress, or DeviceKeyRelease, the *keycode* field is set to the key that caused the change. If *event_type* is ButtonPress or ButtonRelease, *keycode* contains the button number.

When a client receives an XkbControlsNotify event, it can note the changes in a changes structure using *XkbNoteControlsChanges.*

The *wanted* parameter is a bitwise inclusive OR of bits taken from the set of masks specified in Table 1 with "ok" in the *changed_ctrls* column. *XkbNoteControlsChanges* copies any changes reported in *new* and specified in *wanted* into the changes record specified by *old.*

STRUCTURES

The structure for the XkbControlsNotify event is defined as follows:

```
typedef struct {
    int           type;         /* Xkb extension base event code */
    unsigned long serial;       /* X server serial number for event */
    Bool          send_event;   /* True => synthetically generated */
    Display *     display;      /* server connection where event generated */
    Time          time;         /* server time when event generated */
    int           xkb_type;     /* XkbCompatMapNotify */
    int           device;       /* Xkb device ID, will not be XkbUseCoreKbd */
    unsigned int  changed_ctrls; /* bits indicating which controls data have changed*/
    unsigned int  enabled_ctrls; /* controls currently enabled in server */
    unsigned int  enabled_ctrl_changes; /* bits indicating enabled/disabled controls */
    int           num_groups;   /* current number of keyboard groups */
    KeyCode       keycode;      /* != 0 => keycode of key causing change */
    char          event_type;   /* Type of event causing change */
    char          req_major;    /* major event code of event causing change */
    char          req_minor;    /* minor event code of event causing change */
} XkbControlsNotifyEvent;
```

SEE ALSO

XkbSelectEventDetails(3), **XkbSelectEvents**(3)