

NAME

XkbFreeDeviceInfo - Free an XkbDeviceInfoRec structure

SYNOPSIS

```
void XkbFreeDeviceInfo (XkbDeviceInfoPtr device_info, unsigned int which, Bool free_all);
```

ARGUMENTS

device_info

pointer to XkbDeviceInfoRec in which to free items

which

mask of components of device_info to free

free_all

True => free everything, including device_info

DESCRIPTION

If *free_all* is True, the *XkbFreeDeviceInfo* frees all components of *device_info* and the XkbDeviceInfoRec structure pointed to by *device_info* itself. If *free_all* is False, the value of *which* determines which subcomponents are freed. *which* is an inclusive OR of one or more of the values from Table 1. If *which* contains XkbXI_ButtonActionsMask, all button actions associated with *device_info* are freed, *device_info->btn_acts* is set to NULL, and *device_info->num_btns* is set to zero. If *which* contains all bits in XkbXI_IndicatorsMask, all XkbDeviceLedInfoRec structures associated with *device_info* are freed, *device_info->leds* is set to NULL, and *device_info->sz_leds* and *device_info->num_leds* are set to zero. If *which* contains XkbXI_IndicatorMapsMask, all indicator maps associated with *device_info* are cleared, but the number of LEDs and the leds structures themselves is preserved. If *which* contains XkbXI_IndicatorNamesMask, all indicator names associated with *device_info* are cleared, but the number of LEDs and the leds structures themselves is preserved. If *which* contains XkbXI_IndicatorStateMask, the indicator state associated with the *device_info* leds are set to zeros but the number of LEDs and the leds structures themselves is preserved.

Table 1 XkbDeviceInfoRec Mask Bits

Name	XkbDeviceInfoRec Value	Capability If Set
Fields Effected		
XkbXI_KeyboardsMask	(1L <<0)	Clients can use all Xkb requests and events with KeyClass devices supported by the input

device extension.

XkbXI_ButtonActionsMask **num_btns** (1L <<1) Clients can assign key
 btn_acts actions to buttons
 non-KeyClass input
 extension devices.

XkbXI_IndicatorNamesMask **leds->names** (1L <<2) Clients can assign
 names to indicators on
 non-KeyClass input
 extension devices.

XkbXI_IndicatorMapsMask **leds->maps** (1L <<3) Clients can assign
 indicator maps to
 indicators on
 non-KeyClass input
 extension devices.

XkbXI_IndicatorStateMask **leds->state** (1L <<4) Clients can request
 the status of indicators
 on non-KeyClass input
 extension devices.

XkbXI_IndicatorsMask **sz_leds** (0x1c) **XkbXI_IndicatorNamesMask** |
 num_leds **XkbXI_IndicatorMapsMask** |
 leds->* **XkbXI_IndicatorStateMask**

XkbXI_UnsupportedFeaturesMask **unsupported** (1L <<15)

XkbXI_AllDeviceFeaturesMask Those selected (0x1e) **XkbXI_IndicatorsMask** |
 by Value Column **XkbSI_ButtonActionsMask**
 masks

XkbXI_AllFeaturesMask Those selected (0x1f) **XkbSI_AllDeviceFeaturesMask** |
 by Value Column **XkbSI_KeyboardsMask**
 masks

XkbXI_AllDetailsMask Those selected (0x801f) **XkbXI_AllFeaturesMask** |
 by Value column **XkbXI_UnsupportedFeaturesMask**
 masks

STRUCTURES

Information about X Input Extension devices is transferred between a client program and the Xkb extension in an XkbDeviceInfoRec structure:

```
typedef struct {
    char *      name;      /* name for device */
    Atom        type;      /* name for class of devices */
    unsigned short device_spec; /* device of interest */
    Bool        has_own_state; /* True=>this device has its own state */
    unsigned short supported; /* bits indicating supported capabilities */
    unsigned short unsupported; /* bits indicating unsupported capabilities */
    unsigned short num_btns; /* number of entries in btn_acts */
    XkbAction * btn_acts; /* button actions */
    unsigned short sz_leds; /* total number of entries in LEDs vector */
    unsigned short num_leds; /* number of valid entries in LEDs vector */
    unsigned short dflt_kbd_fb; /* input extension ID of default (core kbd) indicator */
    unsigned short dflt_led_fb; /* input extension ID of default indicator feedback */
    XkbDeviceLedInfoPtr leds; /* LED descriptions */
} XkbDeviceInfoRec, *XkbDeviceInfoPtr;
```

```
typedef struct {
    unsigned short led_class; /* class for this LED device */
    unsigned short led_id; /* ID for this LED device */
    unsigned int phys_indicators; /* bits for which LEDs physically present */
    unsigned int maps_present; /* bits for which LEDs have maps in maps */
    unsigned int names_present; /* bits for which LEDs are in names */
    unsigned int state; /* 1 bit => corresponding LED is on */
    Atom names[XkbNumIndicators]; /* names for LEDs */
    XkbIndicatorMapRec maps; /* indicator maps for each LED */
} XkbDeviceLedInfoRec, *XkbDeviceLedInfoPtr;
```