NAME

XkbAddDeviceLedInfo - Initialize an XkbDeviceLedInfoRec structure

SYNOPSIS

XkbDeviceLedInfoPtr XkbAddDeviceLedInfo (XkbDeviceInfoPtr *device_info*, **unsigned int** *led_class*, **unsigned int** *led_id*);

ARGUMENTS

device_info
structure in which to add LED info

led_class
input extension class for LED device of interest

led_id
input extension ID for LED device of interest

DESCRIPTION

XkbAddDeviceLedInfo first checks to see whether an entry matching *led_class* and *led_id* already exists in the *device_info->leds* array. If it finds a matching entry, it returns a pointer to that entry. Otherwise, it checks to be sure there is at least one empty entry in *device_info->leds* and extends it if there is not enough room. It then increments *device_info->num_leds* and fills in the next available entry in *device_info->leds* with *led_class* and *led_id.*

If successful, *XkbAddDeviceLedInfo* returns a pointer to the XkbDeviceLedInfoRec structure that was initialized. If unable to allocate sufficient storage, or if *device_info* points to an invalid XkbDeviceInfoRec structure, or if *led_class* or *led_id* are inappropriate, *XkbAddDeviceLedInfo* returns NULL.

To allocate additional space for button actions in an XkbDeviceInfoRec structure, use *XkbResizeDeviceButtonActions.*

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;
```

SEE ALSO

XkbResizeDeviceButtonActions(3)