NAME

xcb_xkb_get_device_info -

SYNOPSIS

#include <xcb/xkb.h>

Request function

xcb_xkb_get_device_info_cookie_t **xcb_xkb_get_device_info**(xcb_connection_t *_conn_,
xcb_xkb_device_spec_t _deviceSpec_, uint16_t _wanted_, uint8_t _allButtons_, uint8_t _firstButton_,
uint8_t _nButtons_, xcb_xkb_led_class_spec_t _ledClass_, xcb_xkb_id_spec_t _ledID_);

Reply datastructure

```
typedef struct xcb_xkb_get_device_info_reply_t {
    uint8_t    response_type;
    uint8_t    deviceID;
    uint16_t   sequence;
    uint32_t   length;
    uint16_t   present;
    uint16_t   supported;
    uint16_t   unsupported;
    uint16_t   nDeviceLedFBs;
    uint8_t    firstBtnWanted;
    uint8_t    nBtnsWanted;
    uint8_t    firstBtnRtrn;
    uint8_t    nBtnsRtrn;
    uint8_t    totalBtns;
    uint8_t    hasOwnState;
    uint16_t   dfltKbdFB;
    uint16_t   dfltLedFB;
    uint8_t    pad0[2];
    xcb_atom_t devType;
    uint16_t   nameLen;
} xcb_xkb_get_device_info_reply_t;
```

Reply function

xcb_xkb_get_device_info_reply_t ***xcb_xkb_get_device_info_reply**(xcb_connection_t *_conn_,
xcb_xkb_get_device_info_cookie_t _cookie_, xcb_generic_error_t **_e_);

Reply accessors

xcb_xkb_string8_t *__xcb_xkb_get_device_info_name__(const xcb_xkb_get_device_info_request_t *__reply__);

int __xcb_xkb_get_device_info_name_length__(const xcb_xkb_get_device_info_reply_t *__reply__);

xcb_generic_iterator_t __xcb_xkb_get_device_info_name_end__(const xcb_xkb_get_device_info_reply_t *__reply__); uint8_t *xcb_xkb_get_device_info_pad_1 (const xcb_xkb_get_device_info_request_t *reply)

xcb_xkb_action_t *__xcb_xkb_get_device_info_btn_actions__(const xcb_xkb_get_device_info_request_t *__reply__);

int __xcb_xkb_get_device_info_btn_actions_length__(const xcb_xkb_get_device_info_reply_t *__reply__);

xcb_xkb_action_iterator_t __xcb_xkb_get_device_info_btn_actions_iterator__(const xcb_xkb_get_device_info_reply_t *__reply__);

int __xcb_xkb_get_device_info_leds_length__(const xcb_xkb_get_device_info_reply_t *__reply__);

xcb_xkb_device_led_info_iterator_t __xcb_xkb_get_device_info_leds_iterator__(const xcb_xkb_get_device_info_reply_t *__reply__);

REQUEST ARGUMENTS

conn The XCB connection to X11.

deviceSpec TODO: NOT YET DOCUMENTED.

wanted TODO: NOT YET DOCUMENTED.

allButtons TODO: NOT YET DOCUMENTED.

firstButton TODO: NOT YET DOCUMENTED.

nButtons TODO: NOT YET DOCUMENTED.

ledClass TODO: NOT YET DOCUMENTED.

ledID TODO: NOT YET DOCUMENTED.

REPLY FIELDS

response_type The type of this reply, in this case *XCB_XKB_GET_DEVICE_INFO*. This field is also present in the *xcb_generic_reply_t* and can be used to tell replies apart from each other.

sequence The sequence number of the last request processed by the X11 server.

length The length of the reply, in words (a word is 4 bytes).

deviceID TODO: NOT YET DOCUMENTED.

present TODO: NOT YET DOCUMENTED.

supported TODO: NOT YET DOCUMENTED.

unsupported TODO: NOT YET DOCUMENTED.

nDeviceLedFBs TODO: NOT YET DOCUMENTED.

firstBtnWanted TODO: NOT YET DOCUMENTED.

nBtnsWanted TODO: NOT YET DOCUMENTED.

firstBtnRtrn TODO: NOT YET DOCUMENTED.

nBtnsRtrn TODO: NOT YET DOCUMENTED.

totalBtns TODO: NOT YET DOCUMENTED.

hasOwnState TODO: NOT YET DOCUMENTED.

dfltKbdFB TODO: NOT YET DOCUMENTED.

dfltLedFB TODO: NOT YET DOCUMENTED.

devType TODO: NOT YET DOCUMENTED.

nameLen TODO: NOT YET DOCUMENTED.

DESCRIPTION

RETURN VALUE

Returns an *xcb_xkb_get_device_info_cookie_t*. Errors have to be handled when calling the reply function *xcb_xkb_get_device_info_reply*.

If you want to handle errors in the event loop instead, use *xcb_xkb_get_device_info_unchecked*. See **xcb-requests(3)** for details.

ERRORS

This request does never generate any errors.

SEE ALSO

AUTHOR

Generated from xkb.xml. Contact xcb@lists.freedesktop.org for corrections and improvements.