

**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

<i>conn</i>	The XCB connection to X11.
<i>deviceSpec</i>	TODO: NOT YET DOCUMENTED.
<i>wanted</i>	TODO: NOT YET DOCUMENTED.
<i>allButtons</i>	TODO: NOT YET DOCUMENTED.
<i>firstButton</i>	TODO: NOT YET DOCUMENTED.
<i>nButtons</i>	TODO: NOT YET DOCUMENTED.
<i>ledClass</i>	TODO: NOT YET DOCUMENTED.
<i>ledID</i>	TODO: NOT YET DOCUMENTED.

## REPLY FIELDS

<i>response_type</i>	The type of this reply, in this case <i>XCB_XKB_GET_DEVICE_INFO</i> . This field is also present in the <i>xcb_generic_reply_t</i> and can be used to tell replies apart from each other.
<i>sequence</i>	The sequence number of the last request processed by the X11 server.
<i>length</i>	The length of the reply, in words (a word is 4 bytes).
<i>deviceID</i>	TODO: NOT YET DOCUMENTED.
<i>present</i>	TODO: NOT YET DOCUMENTED.
<i>supported</i>	TODO: NOT YET DOCUMENTED.
<i>unsupported</i>	TODO: NOT YET DOCUMENTED.
<i>nDeviceLeds</i>	TODO: NOT YET DOCUMENTED.
<i>firstBtnWanted</i>	TODO: NOT YET DOCUMENTED.
<i>nBtnsWanted</i>	TODO: NOT YET DOCUMENTED.
<i>firstBtnRtrn</i>	TODO: NOT YET DOCUMENTED.
<i>nBtnsRtrn</i>	TODO: NOT YET DOCUMENTED.
<i>totalBtns</i>	TODO: NOT YET DOCUMENTED.
<i>hasOwnState</i>	TODO: NOT YET DOCUMENTED.
<i>dfltKbdFB</i>	TODO: NOT YET DOCUMENTED.
<i>dfltLeds</i>	TODO: NOT YET DOCUMENTED.
<i>devType</i>	TODO: NOT YET DOCUMENTED.
<i>nameLen</i>	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](mailto:xcb@lists.freedesktop.org) for corrections and improvements.