

**NAME**

xcb\_xkb\_list\_components -

**SYNOPSIS**

```
#include <xcb/xkb.h>
```

**Request function**

```
xcb_xkb_list_components_cookie_t xcb_xkb_list_components(xcb_connection_t *conn,
    xcb_xkb_device_spec_t deviceSpec, uint16_t maxNames);
```

**Reply datastructure**

```
typedef struct xcb_xkb_list_components_reply_t {
    uint8_t response_type;
    uint8_t deviceID;
    uint16_t sequence;
    uint32_t length;
    uint16_t nKeymaps;
    uint16_t nKeycodes;
    uint16_t nTypes;
    uint16_t nCompatMaps;
    uint16_t nSymbols;
    uint16_t nGeometries;
    uint16_t extra;
    uint8_t pad0[10];
} xcb_xkb_list_components_reply_t;
```

**Reply function**

```
xcb_xkb_list_components_reply_t *xcb_xkb_list_components_reply(xcb_connection_t *conn,
    xcb_xkb_list_components_cookie_t cookie, xcb_generic_error_t **e);
```

**Reply accessors**

```
int xcb_xkb_list_components_keymaps_length(const xcb_xkb_list_components_reply_t *reply);
```

```
xcb_xkb_listing_iterator_t xcb_xkb_list_components_keymaps_iterator(const
    xcb_xkb_list_components_reply_t *reply);
```

```
int xcb_xkb_list_components_keycodes_length(const xcb_xkb_list_components_reply_t *reply);
```

```
xcb_xkb_listing_iterator_t xcb_xkb_list_components_keycodes_iterator(const
```

```

xcb_xkb_list_components_reply_t *reply);

int xcb_xkb_list_components_types_length(const xcb_xkb_list_components_reply_t *reply);

xcb_xkb_listing_iterator_t xcb_xkb_list_components_types_iterator(const
xcb_xkb_list_components_reply_t *reply);

int xcb_xkb_list_components_compat_maps_length(const xcb_xkb_list_components_reply_t *reply);

xcb_xkb_listing_iterator_t xcb_xkb_list_components_compat_maps_iterator(const
xcb_xkb_list_components_reply_t *reply);

int xcb_xkb_list_components_symbols_length(const xcb_xkb_list_components_reply_t *reply);

xcb_xkb_listing_iterator_t xcb_xkb_list_components_symbols_iterator(const
xcb_xkb_list_components_reply_t *reply);

int xcb_xkb_list_components_geometries_length(const xcb_xkb_list_components_reply_t *reply);

xcb_xkb_listing_iterator_t xcb_xkb_list_components_geometries_iterator(const
xcb_xkb_list_components_reply_t *reply);

```

## REQUEST ARGUMENTS

*conn*            The XCB connection to X11.

*deviceSpec*    TODO: NOT YET DOCUMENTED.

*maxNames*      TODO: NOT YET DOCUMENTED.

## REPLY FIELDS

*response\_type* The type of this reply, in this case *XCB\_XKB\_LIST\_COMPONENTS*. 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.

<i>nKeymaps</i>	TODO: NOT YET DOCUMENTED.
<i>nKeycodes</i>	TODO: NOT YET DOCUMENTED.
<i>nTypes</i>	TODO: NOT YET DOCUMENTED.
<i>nCompatMaps</i>	TODO: NOT YET DOCUMENTED.
<i>nSymbols</i>	TODO: NOT YET DOCUMENTED.
<i>nGeometries</i>	TODO: NOT YET DOCUMENTED.
<i>extra</i>	TODO: NOT YET DOCUMENTED.

**DESCRIPTION****RETURN VALUE**

Returns an *xcb\_xkb\_list\_components\_cookie\_t*. Errors have to be handled when calling the reply function *xcb\_xkb\_list\_components\_reply*.

If you want to handle errors in the event loop instead, use *xcb\_xkb\_list\_components\_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.