

**NAME**

xcb\_randr\_get\_output\_info -

**SYNOPSIS**

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

**Request function**

```
xcb_randr_get_output_info_cookie_t xcb_randr_get_output_info(xcb_connection_t *conn,
    xcb_randr_output_t output, xcb_timestamp_t config_timestamp);
```

**Reply datastructure**

```
typedef struct xcb_randr_get_output_info_reply_t {
    uint8_t    response_type;
    uint8_t    status;
    uint16_t   sequence;
    uint32_t   length;
    xcb_timestamp_t timestamp;
    xcb_randr_crtc_t crtc;
    uint32_t   mm_width;
    uint32_t   mm_height;
    uint8_t    connection;
    uint8_t    subpixel_order;
    uint16_t   num_crtcs;
    uint16_t   num_modes;
    uint16_t   num_preferred;
    uint16_t   num_clones;
    uint16_t   name_len;
} xcb_randr_get_output_info_reply_t;
```

**Reply function**

```
xcb_randr_get_output_info_reply_t *xcb_randr_get_output_info_reply(xcb_connection_t *conn,
    xcb_randr_get_output_info_cookie_t cookie, xcb_generic_error_t **e);
```

**Reply accessors**

```
xcb_randr_crtc_t *xcb_randr_get_output_info_crtcs(const xcb_randr_get_output_info_request_t
    *reply);
```

```
int xcb_randr_get_output_info_crtcs_length(const xcb_randr_get_output_info_reply_t *reply);
```

```

xcb_generic_iterator_t xcb_randr_get_output_info_crtcs_end(const
    xcb_randr_get_output_info_reply_t *reply);

xcb_randr_mode_t *xcb_randr_get_output_info_modes(const xcb_randr_get_output_info_request_t
    *reply);

int xcb_randr_get_output_info_modes_length(const xcb_randr_get_output_info_reply_t *reply);

xcb_generic_iterator_t xcb_randr_get_output_info_modes_end(const
    xcb_randr_get_output_info_reply_t *reply);

xcb_randr_output_t *xcb_randr_get_output_info_clones(const xcb_randr_get_output_info_request_t
    *reply);

int xcb_randr_get_output_info_clones_length(const xcb_randr_get_output_info_reply_t *reply);

xcb_generic_iterator_t xcb_randr_get_output_info_clones_end(const
    xcb_randr_get_output_info_reply_t *reply);

uint8_t *xcb_randr_get_output_info_name(const xcb_randr_get_output_info_request_t *reply);

int xcb_randr_get_output_info_name_length(const xcb_randr_get_output_info_reply_t *reply);

xcb_generic_iterator_t xcb_randr_get_output_info_name_end(const
    xcb_randr_get_output_info_reply_t *reply);

```

## REQUEST ARGUMENTS

*conn*            The XCB connection to X11.

*output*            TODO: NOT YET DOCUMENTED.

*config\_timestamp*  
                  TODO: NOT YET DOCUMENTED.

## REPLY FIELDS

*response\_type*    The type of this reply, in this case *XCB\_RANDR\_GET\_OUTPUT\_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.

<i>length</i>	The length of the reply, in words (a word is 4 bytes).
<i>status</i>	TODO: NOT YET DOCUMENTED.
<i>timestamp</i>	TODO: NOT YET DOCUMENTED.
<i>crtc</i>	TODO: NOT YET DOCUMENTED.
<i>mm_width</i>	TODO: NOT YET DOCUMENTED.
<i>mm_height</i>	TODO: NOT YET DOCUMENTED.
<i>connection</i>	TODO: NOT YET DOCUMENTED.
<i>subpixel_order</i>	TODO: NOT YET DOCUMENTED.
<i>num_crtcs</i>	TODO: NOT YET DOCUMENTED.
<i>num_modes</i>	TODO: NOT YET DOCUMENTED.
<i>num_preferred</i>	TODO: NOT YET DOCUMENTED.
<i>num_clones</i>	TODO: NOT YET DOCUMENTED.
<i>name_len</i>	TODO: NOT YET DOCUMENTED.

## DESCRIPTION

## RETURN VALUE

Returns an *xcb\_randr\_get\_output\_info\_cookie\_t*. Errors have to be handled when calling the reply function *xcb\_randr\_get\_output\_info\_reply*.

If you want to handle errors in the event loop instead, use *xcb\_randr\_get\_output\_info\_unchecked*. See **xcb-requests(3)** for details.

## ERRORS

This request does never generate any errors.

## SEE ALSO

## AUTHOR

Generated from randr.xml. Contact [xcb@lists.freedesktop.org](mailto:xcb@lists.freedesktop.org) for corrections and improvements.