

NAME

xcb_randr_get_crtc_transform -

SYNOPSIS

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

Request function

```
xcb_randr_get_crtc_transform_cookie_t xcb_randr_get_crtc_transform(xcb_connection_t *conn,
    xcb_randr_crtc_t crtc);
```

Reply datastructure

```
typedef struct xcb_randr_get_crtc_transform_reply_t {
    uint8_t      response_type;
    uint8_t      pad0;
    uint16_t     sequence;
    uint32_t     length;
    xcb_render_transform_t pending_transform;
    uint8_t      has_transforms;
    uint8_t      pad1[3];
    xcb_render_transform_t current_transform;
    uint8_t      pad2[4];
    uint16_t     pending_len;
    uint16_t     pending_nparams;
    uint16_t     current_len;
    uint16_t     current_nparams;
} xcb_randr_get_crtc_transform_reply_t;
```

Reply function

```
xcb_randr_get_crtc_transform_reply_t *xcb_randr_get_crtc_transform_reply(xcb_connection_t *conn,
    xcb_randr_get_crtc_transform_cookie_t cookie, xcb_generic_error_t **e);
```

Reply accessors

```
char *xcb_randr_get_crtc_transform_pending_filter_name(const
    xcb_randr_get_crtc_transform_request_t *reply);
```

```
int xcb_randr_get_crtc_transform_pending_filter_name_length(const
    xcb_randr_get_crtc_transform_reply_t *reply);
```

```
xcb_generic_iterator_t xcb_randr_get_crtc_transform_pending_filter_name_end(const
```

```
xcb_randr_get_crtc_transform_reply_t *reply); uint8_t *xcb_randr_get_crtc_transform_pad_3
(const xcb_randr_get_crtc_transform_request_t *reply)
```

```
xcb_render_fixed_t *xcb_randr_get_crtc_transform_pending_params(const
xcb_randr_get_crtc_transform_request_t *reply);
```

```
int xcb_randr_get_crtc_transform_pending_params_length(const
xcb_randr_get_crtc_transform_reply_t *reply);
```

```
xcb_generic_iterator_t xcb_randr_get_crtc_transform_pending_params_end(const
xcb_randr_get_crtc_transform_reply_t *reply);
```

```
char *xcb_randr_get_crtc_transform_current_filter_name(const
xcb_randr_get_crtc_transform_request_t *reply);
```

```
int xcb_randr_get_crtc_transform_current_filter_name_length(const
xcb_randr_get_crtc_transform_reply_t *reply);
```

```
xcb_generic_iterator_t xcb_randr_get_crtc_transform_current_filter_name_end(const
xcb_randr_get_crtc_transform_reply_t *reply); uint8_t *xcb_randr_get_crtc_transform_pad_4
(const xcb_randr_get_crtc_transform_request_t *reply)
```

```
xcb_render_fixed_t *xcb_randr_get_crtc_transform_current_params(const
xcb_randr_get_crtc_transform_request_t *reply);
```

```
int xcb_randr_get_crtc_transform_current_params_length(const xcb_randr_get_crtc_transform_reply_t
*reply);
```

```
xcb_generic_iterator_t xcb_randr_get_crtc_transform_current_params_end(const
xcb_randr_get_crtc_transform_reply_t *reply);
```

REQUEST ARGUMENTS

conn The XCB connection to X11.

crtc TODO: NOT YET DOCUMENTED.

REPLY FIELDS

response_type The type of this reply, in this case `XCB_RANDR_GET_CRTC_TRANSFORM`. 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).

pending_transform
 TODO: NOT YET DOCUMENTED.

has_transforms TODO: NOT YET DOCUMENTED.

current_transform
 TODO: NOT YET DOCUMENTED.

pending_len TODO: NOT YET DOCUMENTED.

pending_nparams
 TODO: NOT YET DOCUMENTED.

current_len TODO: NOT YET DOCUMENTED.

current_nparams
 TODO: NOT YET DOCUMENTED.

DESCRIPTION

RETURN VALUE

Returns an *xcb_randr_get_crtc_transform_cookie_t*. Errors have to be handled when calling the reply function *xcb_randr_get_crtc_transform_reply*.

If you want to handle errors in the event loop instead, use *xcb_randr_get_crtc_transform_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 for corrections and improvements.