**NAME**

xcb_get_property - Gets a window property

**SYNOPSIS**

**#include <xcb/xproto.h>**

**Request function**

xcb_get_property_cookie_t **xcb_get_property**(xcb_connection_t *_conn_, uint8_t _\_delete_,
xcb_window_t _window_, xcb_atom_t _property_, xcb_atom_t _type_, uint32_t _long_offset_,
uint32_t _long_length_);

**Reply datastructure**

typedef struct xcb_get_property_reply_t {
uint8_t    _response_type_;
uint8_t    _format_;
uint16_t   _sequence_;
uint32_t   _length_;
xcb_atom_t _type_;
uint32_t   _bytes_after_;
uint32_t   _value_len_;
uint8_t    _pad0_[12];
} **xcb_get_property_reply_t**;

**Reply function**

xcb_get_property_reply_t ***xcb_get_property_reply**(xcb_connection_t *_conn_,
xcb_get_property_cookie_t _cookie_, xcb_generic_error_t **_e_);

**Reply accessors**

void ***xcb_get_property_value**(const xcb_get_property_request_t *_reply_);

int **xcb_get_property_value_length**(const xcb_get_property_reply_t *_reply_);

xcb_generic_iterator_t **xcb_get_property_value_end**(const xcb_get_property_reply_t *_reply_);

**REQUEST ARGUMENTS**

_conn_            The XCB connection to X11.

_\_delete_         Whether the property should actually be deleted. For deleting a property, the specified
                  _type_ has to match the actual property type.

*window*        The window whose property you want to get.

*property*        The property you want to get (an atom).

*type*        The type of the property you want to get (an atom).

*long_offset*        Specifies the offset (in 32-bit multiples) in the specified property where the data is to
be retrieved.

*long_length*        Specifies how many 32-bit multiples of data should be retrieved (e.g. if you set
*long_length* to 4, you will receive 16 bytes of data).

**REPLY FIELDS**

*response_type*        The type of this reply, in this case *XCB_GET_PROPERTY*. 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).

*format*        Specifies whether the data should be viewed as a list of 8-bit, 16-bit, or 32-bit
quantities. Possible values are 8, 16, and 32. This information allows the X server to
correctly perform byte-swap operations as necessary.

*type*        The actual type of the property (an atom).

*bytes_after*        The number of bytes remaining to be read in the property if a partial read was
performed.

*value_len*        The length of value. You should use the corresponding accessor instead of this field.

**DESCRIPTION**

Gets the specified *property* from the specified *window*. Properties are for example the window title
(*WM_NAME*) or its minimum size (*WM_NORMAL_HINTS*).  Protocols such as EWMH also use
properties - for example EWMH defines the window title, encoded as UTF-8 string, in the
*_NET_WM_NAME* property.

TODO: talk about *type*

TODO: talk about *delete*

TODO: talk about the offset/length thing. what's a valid use case?

**RETURN VALUE**

Returns an *xcb_get_property_cookie_t*. Errors have to be handled when calling the reply function *xcb_get_property_reply*.

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

**ERRORS**

*xcb_atom_error_t*

                *property* or *type* do not refer to a valid atom.

*xcb_value_error_t*

                The specified *long_offset* is beyond the actual property length (e.g. the property has a length of 3 bytes and you are setting *long_offset* to 1, resulting in a byte offset of 4).

*xcb_window_error_t*

                The specified *window* does not exist.

**EXAMPLE**

```
/*
 * Prints the WM_NAME property of the window.
 *
 */
void my_example(xcb_connection_t *c, xcb_window_t window) {
    xcb_get_property_cookie_t cookie;
    xcb_get_property_reply_t *reply;

    /* These atoms are predefined in the X11 protocol. */
    xcb_atom_t property = XCB_ATOM_WM_NAME;
    xcb_atom_t type = XCB_ATOM_STRING;

    // TODO: a reasonable long_length for WM_NAME?
    cookie = xcb_get_property(c, 0, window, property, type, 0, 0);
    if ((reply = xcb_get_property_reply(c, cookie, NULL))) {
        int len = xcb_get_property_value_length(reply);
        if (len == 0) {
            printf("TODO\n");
```

```
            free(reply);
            return;
        }
        printf("WM_NAME is %.*s\n", len,
            (char*)xcb_get_property_value(reply));
    }
    free(reply);
}
```

**SEE ALSO**

    **xcb-requests**(3), **xcb-examples**(3), **xcb_intern_atom**(3), **xprop**(1)

**AUTHOR**

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