NAME

xcb-requests - about request manpages

DESCRIPTION

Every request in X11, like *MapWindow*, corresponds to a number of functions and data structures in XCB. For *MapWindow*, XCB provides the function *xcb_map_window*, which fills the *xcb_map_window_request_t* data structure and writes that to the X11 connection. Since the *MapWindow* request does not have a reply, this is the most simple case.

REPLIES

Many requests have replies. For each reply, XCB provides at least a corresponding data structure and a function to return a pointer to a filled data structure. Let's take the *InternAtom* request as an example: XCB provides the *xcb_intern_atom_reply_t* data structure and *xcb_intern_atom_reply* function. For replies which are more complex (for example lists, such as in *xcb_list_fonts*), accessor functions are provided.

COOKIES

XCB returns a cookie for each request you send. This is an XCB-specific data structure containing the sequence number with which the request was sent to the X11 server. To get any reply, you have to provide that cookie (so that XCB knows which of the waiting replies you want). Here is an example to illustrate the use of cookies:

```
void my_example(xcb_connection *conn) {
   xcb_intern_atom_cookie_t cookie;
   xcb_intern_atom_reply_t *reply;

   cookie = xcb_intern_atom(conn, 0, strlen("_NET_WM_NAME"), "_NET_WM_NAME");
   /* ... do other work here if possible ... */
   if ((reply = xcb_intern_atom_reply(conn, cookie, NULL))) {
      printf("The _NET_WM_NAME atom has ID %u\n", reply->atom);
   }
   free(reply);
}
```

CHECKED VS. UNCHECKED

The checked and unchecked suffixes for functions determine which kind of error handling is used for

this specific request.

For requests which have no reply (for example *xcb_map_window*), errors will be delivered to the event loop (you will receive an X11 event of type 0 when calling *xcb_poll_for_event*). If you want to explicitly check for errors in a blocking fashion, call the _checked version of the function (for example *xcb_map_window_checked*) and use *xcb_request_check*.

For requests which have a reply (for example *xcb_intern_atom*), errors will be checked when calling the reply function. To get errors in the event loop instead, use the _unchecked version of the function (for example *xcb_intern_atom_unchecked*).

Here is an example which illustrates the four different ways of handling errors:

```
/*
 * Request without a reply, handling errors in the event loop (default)
 *
 */
void my_example(xcb_connection *conn, xcb_window_t window) {
    /* This is a request without a reply. Errors will be delivered to the event
     * loop. Getting an error to xcb_map_window most likely is a bug in our
     * program, so we don't need to check for that in a blocking way. */
    xcb_map_window(conn, window);

    /* ... of course your event loop would not be in the same function ... */
    while ((event = xcb_wait_for_event(conn)) != NULL) {
        if (event->response_type == 0) {
            fprintf("Received X11 error %d\n", error->error_code);
            free(event);
            continue;
        }

        /* ... handle a normal event ... */
    }
}

/*
 * Request without a reply, handling errors directly
 *
 */
```

```
void my_example(xcb_connection *conn, xcb_window_t deco, xcb_window_t window) {
   /* A reparenting window manager wants to know whether a new window was
    * successfully reparented. If not (because the window got destroyed
    * already, for example), it does not make sense to map an empty window
    * decoration at all, so we need to know this right now. */
   xcb_void_cookie_t cookie = xcb_reparent_window_checked(conn, window,
                                                          deco, 0, 0);
   xcb_generic_error_t *error;
   if ((error = xcb_request_check(conn, cookie))) {
      fprintf(stderr, "Could not reparent the window\n");
      free(error);
      return;
   }

   /* ... do window manager stuff here ... */
}

/*
 * Request with a reply, handling errors directly (default)
 *
 */
void my_example(xcb_connection *conn, xcb_window_t window) {
   xcb_intern_atom_cookie_t cookie;
   xcb_intern_atom_reply_t *reply;
   xcb_generic_error_t *error;

   cookie = xcb_intern_atom(c, 0, strlen("_NET_WM_NAME"), "_NET_WM_NAME");
   /* ... do other work here if possible ... */
   if ((reply = xcb_intern_atom_reply(c, cookie, &error))) {
      printf("The _NET_WM_NAME atom has ID %u\n", reply->atom);
      free(reply);
   } else {
      fprintf(stderr, "X11 Error %d\n", error->error_code);
      free(error);
   }
}

/*
 * Request with a reply, handling errors in the event loop
 *
```

```
 */
void my_example(xcb_connection *conn, xcb_window_t window) {
   xcb_intern_atom_cookie_t cookie;
   xcb_intern_atom_reply_t *reply;

   cookie = xcb_intern_atom_unchecked(c, 0, strlen("_NET_WM_NAME"),
                                      "_NET_WM_NAME");
   /* ... do other work here if possible ... */
   if ((reply = xcb_intern_atom_reply(c, cookie, NULL))) {
      printf("The _NET_WM_NAME atom has ID %u\n", reply->atom);
      free(reply);
   }

   /* ... of course your event loop would not be in the same function ... */
   while ((event = xcb_wait_for_event(conn)) != NULL) {
      if (event->response_type == 0) {
         fprintf("Received X11 error %d\n", error->error_code);
         free(event);
         continue;
      }

      /* ... handle a normal event ... */
   }
}
```

SEE ALSO

xcb_map_window(3), **xcb_intern_atom**(3), **xcb_list_fonts**(3), **xcb_poll_for_event**(3), **xcb_request_check**(3)

AUTHOR

Michael Stapelberg <michael+xcb at stapelberg dot de>