NAME

xcb_change_gc - change graphics context components

SYNOPSIS

#include <xcb/xproto.h>

Request function

xcb_void_cookie_t **xcb_change_gc**(xcb_connection_t *conn*, xcb_gcontext_t *gc*, uint32_t *value_mask*, const void **value_list*);

REQUEST ARGUMENTS

conn   The XCB connection to X11.

gc   The graphics context to change.

value_mask   One of the following values:

XCB_GC_FUNCTION
                  TODO: Refer to GX

XCB_GC_PLANE_MASK
                  In graphics operations, given a source and destination pixel, the result
                  is computed bitwise on corresponding bits of the pixels; that is, a
                  Boolean operation is performed in each bit plane. The plane-mask
                  restricts the operation to a subset of planes, so the result is:

                  ((src FUNC dst) AND plane-mask) OR (dst AND (NOT plane-
                  mask))

XCB_GC_FOREGROUND
                  Foreground colorpixel.

XCB_GC_BACKGROUND
                  Background colorpixel.

XCB_GC_LINE_WIDTH
                  The line-width is measured in pixels and can be greater than or equal
                  to one, a wide line, or the special value zero, a thin line.

XCB_GC_LINE_STYLE

The line-style defines which sections of a line are drawn: Solid The full path of the line is drawn. DoubleDash   The full path of the line is drawn, but the even dashes are filled differently

than the odd dashes (see fill-style), with Butt cap-style used where even and

odd dashes meet. OnOffDash   Only the even dashes are drawn, and cap-style applies to all internal ends of

the individual dashes (except NotLast is treated as Butt).

XCB_GC_CAP_STYLE

The cap-style defines how the endpoints of a path are drawn: NotLast The result is equivalent to Butt, except that for a line-width of zero the final

endpoint is not drawn. Butt   The result is square at the endpoint (perpendicular to the slope of the line)

with no projection beyond. Round   The result is a circular arc with its diameter equal to the line-width, centered

on the endpoint; it is equivalent to Butt for line-width zero. Projecting The result is square at the end, but the path continues beyond the endpoint for

a distance equal to half the line-width; it is equivalent to Butt for line-width

zero.

XCB_GC_JOIN_STYLE

The join-style defines how corners are drawn for wide lines: Miter The outer edges of the two lines extend to meet at an angle. However, if the

angle is less than 11 degrees, a Bevel join-style is used instead. Round   The result is a circular arc with a diameter equal to the line-width, centered

on the joinpoint. Bevel   The result is Butt endpoint styles, and then the triangular notch is filled.

XCB_GC_FILL_STYLE

The fill-style defines the contents of the source for line, text, and fill requests. For all text and fill requests (for example, PolyText8, PolyText16, PolyFillRectangle, FillPoly, and PolyFillArc) as well as for line requests with line-style Solid, (for example, PolyLine, PolySegment, PolyRectangle, PolyArc) and for the even dashes for

line requests with line-style OnOffDash or DoubleDash: Solid Foreground Tiled   Tile OpaqueStippled   A tile with the same width and height as stipple but with background   everywhere stipple has a zero and with foreground everywhere stipple

has a one Stippled   Foreground masked by stipple For the odd dashes for line requests with line-style DoubleDash: Solid   Background Tiled   Same as for even dashes OpaqueStippled   Same as for even dashes Stippled   Background masked by stipple

XCB_GC_FILL_RULE

XCB_GC_TILE

The tile/stipple represents an infinite two-dimensional plane with the tile/stipple replicated in all dimensions. When that plane is superimposed on the drawable for use in a graphics operation, the upper-left corner of some instance of the tile/stipple is at the coordinates within the drawable specified by the tile/stipple origin. The tile/stipple and clip origins are interpreted relative to the origin of whatever destination drawable is specified in a graphics request. The tile pixmap must have the same root and depth as the gcontext (or a Match error results). The stipple pixmap must have depth one and must have the same root as the gcontext (or a Match error results). For fill-style Stippled (but not fill-style OpaqueStippled), the stipple pattern is tiled in a single plane and acts as an additional clip mask to be ANDed with the clip-mask. Any size pixmap can be used for tiling or stippling, although some sizes may be faster to use than others.

XCB_GC_STIPPLE

The tile/stipple represents an infinite two-dimensional plane with the tile/stipple replicated in all dimensions. When that plane is superimposed on the drawable for use in a graphics operation, the upper-left corner of some instance of the tile/stipple is at the coordinates within the drawable specified by the tile/stipple origin. The tile/stipple and clip origins are interpreted relative to the origin of whatever destination drawable is specified in a graphics request. The tile pixmap must have the same root and depth as the gcontext (or a Match error results). The stipple pixmap must have depth one and

must have the same root as the gcontext (or a Match error results). For
fill-style Stippled (but not fill-style OpaqueStippled), the stipple
pattern is tiled in a single plane and acts as an additional clip mask to
be ANDed with the clip-mask. Any size pixmap can be used for tiling
or stippling, although some sizes may be faster to use than others.

XCB_GC_TILE_STIPPLE_ORIGIN_X
                     TODO

XCB_GC_TILE_STIPPLE_ORIGIN_Y
                     TODO

XCB_GC_FONT
                     Which font to use for the *ImageText8* and *ImageText16* requests.

XCB_GC_SUBWINDOW_MODE
                     For ClipByChildren, both source and destination windows are
                     additionally clipped by all viewable InputOutput children. For
                     IncludeInferiors, neither source nor destination window is clipped by
                     inferiors. This will result in including subwindow contents in the
                     source and drawing through subwindow boundaries of the destination.
                     The use of IncludeInferiors with a source or destination window of
                     one depth with mapped inferiors of differing depth is not illegal, but
                     the semantics is undefined by the core protocol.

XCB_GC_GRAPHICS_EXPOSURES
                     Whether ExposureEvents should be generated (1) or not (0).

                     The default is 1.

XCB_GC_CLIP_ORIGIN_X
                     TODO

XCB_GC_CLIP_ORIGIN_Y
                     TODO

XCB_GC_CLIP_MASK
                     The clip-mask restricts writes to the destination drawable. Only pixels
                     where the clip-mask has bits set to 1 are drawn. Pixels are not drawn
                     outside the area covered by the clip-mask or where the clip-mask has

bits set to 0. The clip-mask affects all graphics requests, but it does not clip sources. The clip-mask origin is interpreted relative to the origin of whatever destination drawable is specified in a graphics request. If a pixmap is specified as the clip-mask, it must have depth 1 and have the same root as the gcontext (or a Match error results). If clip-mask is None, then pixels are always drawn, regardless of the clip origin. The clip-mask can also be set with the SetClipRectangles request.

XCB_GC_DASH_OFFSET
            TODO

XCB_GC_DASH_LIST
            TODO

XCB_GC_ARC_MODE
            TODO

value_list   Values for each of the components specified in the bitmask *value_mask*. The order has to correspond to the order of possible *value_mask* bits. See the example.

DESCRIPTION
Changes the components specified by *value_mask* for the specified graphics context.

RETURN VALUE
Returns an *xcb_void_cookie_t*. Errors (if any) have to be handled in the event loop.

If you want to handle errors directly with *xcb_request_check* instead, use *xcb_change_gc_checked*. See **xcb-requests(3)** for details.

ERRORS
xcb_alloc_error_t
            The X server could not allocate the requested resources (no memory?).

xcb_font_error_t
            TODO: reasons?

xcb_g_context_error_t
            TODO: reasons?

  xcb_match_error_t
      TODO: reasons?

  xcb_pixmap_error_t
      TODO: reasons?

  xcb_value_error_t
      TODO: reasons?

EXAMPLE

```
/*
 * Changes the foreground color component of the specified graphics context.
 *
 */
void my_example(xcb_connection_t *conn, xcb_gcontext_t gc, uint32_t fg, uint32_t bg) {
    /* C99 allows us to use a compact way of changing a single component: */
    xcb_change_gc(conn, gc, XCB_GC_FOREGROUND, (uint32_t[]){ fg });

    /* The more explicit way. Beware that the order of values is important! */
    uint32_t mask = 0;
    mask |= XCB_GC_FOREGROUND;
    mask |= XCB_GC_BACKGROUND;

    uint32_t values[] = {
        fg,
        bg
    };
    xcb_change_gc(conn, gc, mask, values);
    xcb_flush(conn);
}
```

SEE ALSO
  xcb-requests(3), **xcb-examples**(3)

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