:width, **:height** — **card16** values defining the size of the image returned.These arguments are required.

:plane-mask — A pixel mask.

:format — Either **:xy–pixmap** or **:z–pixmap**. This argument is required.

:result-type — The type of image data sequence to return.

put-raw-image *drawable gcontext data* &key (**:start** 0) **:depth :x :y** **:width :height** (**:left-pad** 0) **:format** Function

Displays a region of the image data defined by **:start**, **:left-pad**, **:width,** and **:height** on the destination *drawable*, with the upper-left pixel of the image region displayed at the *drawable* position given by **:x** and **:y**.

The **:format** can be either **:xy-pixmap**, **:z-pixmap**, or **:bitmap**. If **:xy-pixmap** or **:z-pixmap** formats are used, **:depth** must match the depth of the destination *drawable*. For **:xy-pixmap**, the data must be in XY format. For **:z-pixmap**, the data must be in Z format for the given **:depth**.

If the **:format** is **:bitmap**, the **:depth** must be 1. In this case, the image is combined with the foreground and background pixels of the *gcontext*. 1 bits of the image are displayed in the foreground pixel and 0 bits are displayed in the background pixel.

The **:left-pad** must be zero for **:z-pixmap** format. For **:bitmap** and **:xy-pixmap** formats, the **:left-pad** must be less than the bitmap-scanline-pad for the *drawable*'s display (see **display-bitmap-format**). The first **:left-pad** bits in every scanline are to be ignored by the server; the actual image begins that many bits into the data.

The following attributes of the *gcontext* are used to display the **image**: clip-mask, clip-x, clip-y, function, plane-mask, and subwindow-mode.

The calling program is responsible for handling the byte-order and bit-order required by the server for the *drawable*'s display (see **display-byte-order** and **display-image-lsb-first-p**).

drawable — The destination **drawable**.

gcontext — The graphics context used to display the image.

data — A sequence of integers.

:start — The index of the first element of *data* displayed.

:depth — The number of bits per pixel displayed. This argument is required.

:x, **:y** — The position in the *drawable* where the image region is displayed. These arguments are required.

:width, **:height** — **card16** values defining the size of the image region displayed. These arguments are required.

:left-pad — A **card8** specifying the number of leading bits to discard for each image scanline.

:format — One of **:bitmap**, **:xy-pixmap**, or **:z-pixmap**.

Image Files

7.4 CLX provides functions that allow images to be written to a file in a standard X format. The following paragraphs describe the CLX functions used to:

- Read an image from a file.
- Write an image to a file.

read-bitmap-file *pathname* Function

Returns:
image — Type **image**.

Reads an image file in standard X format and returns an **image** object. The returned *image* can have depth greater than one.

pathname — An image file pathname.

write-bitmap-file *pathname image* &optional *name* Function

Writes the *image* to an image file in standard X format. The *image* can have depth greater than one. The *name* is an image identifier written to the file; the default *name* is (**or** (**image-name** *image*) **'image**).

pathname — An image file pathname.

image — An **image** object.

name — A **stringable** image name.

Direct Image Transfer

7.5 For cases where the **image** representation is not needed, CLX provides functions to read and display image data directly.

get-raw-image *drawable* &key **:data** (**:start** 0) **:x :y :width :height** **:plane-mask** (**:format :z-format**) (**:result-type '**(**vector card8**)) Function

Returns:
data — Type **sequence** or **card8**.
depth — Type **card8**.
visual — Type **card29**.

Returns a sequence of image data from the region of the *drawable* given by **:x**, **:y**, **:width**, and **:height**. If **:data** is given, it is modified beginning with the element at the **:start** index and returned. The *depth* and *visua*l type ID of the *drawable* are also returned.

The bits for all planes selected by 1 bits in the **:plane-mask** are returned as zero; the default **:plane-mask** is all 1 bits. The **:format** of the returned pixel values may be either **:xy-format** or **:z-format**. The **:result-type** defines the type of image data returned.

The calling program is responsible for handling the byte-order and bit-order returned by the server for the *drawable*'s display (see **display-byte-order** and **display-image-lsb-first-p**).

drawable — A **drawable**.

:data — An optional **sequence** of **card8**.

:start — The index of the first **:data** element modified.

:x, **:y** — **card16** values defining the size of the **image** returned. These arguments are required.

The **:result-type** defines the type of image object returned:

- **'image-x** — A basic **image** object is returned.
- **'image-xy** — An **image-xy** is returned.
- **'image-z** — An **image-z** is returned.

By default, **:result-type** is **'image-z** if **:format** is **:z-format** and **'image-xy** if **:format** is **:xy-format**.

drawable — A **drawable**.

:x, **:y** — **card16** values defining the upper-left **drawable** pixel returned. These arguments are required.

:width, **:height** — **card16** values defining the size of the *image* returned. These arguments are required.

:plane-mask — A pixel mask.

:format — Either **:xy-pixmap** or **:z-pixmap**.

:result-type — One of **'image-x**, **'image-xy**, or **'image-z**.

put-image *drawable gcontext image* &key (**:src-x** 0) (**:src-y** 0) **:x :y** **:width :height :bitmap-p** Function

Displays a region of the *image* defined by **:src-x**, **:src-y**, **:width**, and **:height** on the destination d*rawable*, with the upper-left pixel of the *image* region displayed at the *drawable* position given by **:x** and **:y**. By default, **:width** is:

(– (**image-width** *image*) **:src-x**)

and **:height** is:

(– (**image-height** *image*) **:src-y**)

The following attributes of the *gcontext* are used to display the **image**: clip-mask, clip-x, clip-y, function, plane-mask, and subwindow-mode.

The **:bitmap-p** argument applies only to images of depth 1. In this case, if **:bitmap-p** is true or if the *image* is a basic **image** object created with **:format :bitmap**, the *image* is combined with the foreground and background pixels of the **gcontext**. 1 bits of the *image* are displayed in the foreground pixel and 0 bits are displayed in the background pixel.

drawable — The destination **drawable**.

gcontext — The graphics context used to display the *image*.

image — An **image** object.

:src-x, **:src-y** — **card16** values defining the upper-left position of the *image* region to display.

:x, **:y** — The position in the *drawable* where the *image* region is displayed. These arguments are required.

:width, **:height** — **card16** values defining the size of the *image* region displayed.

:bitmap-p — If *image* is depth 1, then if true, foreground and background pixels are used to display 1 and 0 bits of the *image*.

copy-image *image* &key (**:x** 0) (**:y** 0) **:width :height :result-type** Function

Returns:
new-image — Type **image**.

Returns a new image, of the given **:result-type**, containing a copy of the portion of the *image* defined by **:x**, **:y**, **:width**, and **:height**. By default, **:width** is:

(– (**image-width** *image*) **:x**)

and **:height** is:

(– (**image-height** *image*) **:y**)

If necessary, the new image is converted to the **:result-type**, that can be one of the following values:

- **'image-x** — A basic **image** object is returned.
- **'image-xy** — An **image-xy** is returned.
- **'image-z** — An **image-z** is returned.

image — An **image** object.

:x, **:y** — **card16** values defining the position of the upper-left corner of the subimage copied.

:width, **:height** — **card16** values defining the size of subimage copied.

:result-type — One of **'image-x**, **'image-xy**, or **'image-z**.

get-image *drawable* &key **:x :y :width :height :plane-mask** Function
(**:format :z-format**) **:result-type**

Returns:
image — Type **image**.

Returns an *image* containing pixel values from the region of the *drawable* given by **:x**, **:y**, **:width**, and **:height**. The bits for all planes selected by 1 bits in the **:plane-mask** are returned as zero; the default **:plane-mask** is all 1 bits. The **:format** of the returned pixel values may be either **:xy-format** or **:z-format**.

If the **:data** is a list, each element must be a bitmap of equal size. **:width** and **:height** default to the bitmap width — (**array-dimension bitmap** 1) — and the bitmap height — (**array-dimension bitmap** 0) — respectively. **:depth** defaults to the number of bitmaps.

If the **:data** is a **pixarray**, **:width** and **:height** default to the **pixarray** width — (**array-dimension pixarray** 1), and the pixarray height — (**array-dimension pixarray** 0), respectively. **:depth** defaults to (**pixarray-depth :data**). The **:bits-per-pixel** is rounded to a valid size, if necessary. By default, the **:bits-per-pixel** is equal to the **:depth**.

If the **:data** is an array of **card8**, the **:width** and **:height** are required to interpret the image data correctly. The **:bits-per-pixel** defaults to the **:depth**, and the **:depth** defaults to 1. **:bytes-per-line** defaults to:

(**floor** (**length :data**) (* **:bits-per-pixel :height**))

The **:format** defines the storage format of image data bytes and can be one of the following values:

- **:xy-pixmap** — The **:data** is organized as a set of bitmaps representing image bit planes, appearing in most-significant to least-significant bit order.
- **:z-pixmap** — The **:data** is organized as a set of pixel values in scanline order.
- **:bitmap** — Similar to **:xy-pixmap**, except that the **:depth** must be 1, and 1 and 0 bits represent the foreground and background pixels, respectively.

By default, the **:format** is **:bitmap** if **:depth** is 1; otherwise, **:z-pixmap**.

:bit-lsb-first-p — For a returned image, true if the order of bits in each **:data** byte is least-significant bit first.

:bits-per-pixel — One of 1, 4, 8, 16, 24, or 32.

:blue-mask — For **:true-color** or **:direct-color** images, a pixel mask.

:byte-lsb-first-p — For a returned *image*, true if the **:data** byte order is least-significant byte first.

:bytes-per-line — For a returned *image*, the number of **:data** bytes per scanline.

:data — Either a **list** of **bitmaps**, a **pixarray**, or an array of **card8** bytes.

:depth — The number of bits per displayed pixel.

:format — One of **:bitmap**, **:xy-format**, or **:z-format**.

:green-mask — For **:true-color** or **:direct-color** images, a pixel mask.

:height — A **card16** for the image height in pixels.

:name — An optional **stringable** for the image name.

:plist — An optional image property list.

:red-mask — For **:true-color** or **:direct-color** images, a pixel mask.

:width — A **card16** for the image width in pixels.

:x-hot — For a **cursor** image, the x position of the hot spot.

:y-hot — For a cursor image, the y position of the hot spot.

XY-Format Images

7.2.2 The **image-xy** subtype represents an image as a sequence of bitmaps, one for each plane of the image, in most-significant to least-significant bit order. The following paragraphs describe the additional CLX functions that can be used to access **image-xy** objects.

image-xy-bitmap-list *image* Function

Returns:
bitmaps — Type **list** of **bitmap**.

Returns and (with **setf**) changes the list of bitmap planes for the *image*.

image — An **image-xy** object.

Z-Format Images

7.2.3 The **image-z** subtype represents an image as a two-dimensional array of pixels, in scanline order. The following paragraphs describe the additional CLX functions that can be used to access **image-z** objects.

image-z-bits-per-pixel *image* Function

Returns:
pixel-data-size — One of 1, 4, 8, 16, 24, or 32.

Returns and (with **setf**) changes the number of bits per data unit used to contain a pixel value for the *image*. Depending on the storage format for image data, this value can be larger than the actual *image* depth.

image — An **image-z** object.

image-z-pixarray *image* Function

Returns:
pixarray — Type **pixarray**.

Returns and (with **setf**) changes the two-dimensional array of pixel data for the *image*.

image — An **image-z** object.

Image Functions

7.3 The following paragraphs describe the CLX functions used to:

- Create an **image** object.
- Copy an image or a subimage.
- Read an image from a **drawable**.
- Display an image to a **drawable**.

create-image &key **:bit-lsb-first-p :bits-per-pixel :blue-mask** Function
:byte-lsb-first-p :bytes-per-line :data :depth :format
:green-mask :height :name :plist :red-mask :width
:x-hot :y-hot

Returns:
image — Type **image**.

Creates an **image** object from the given **:data** and returns either an **image**, **image-xy**, or an **image-z**, depending on the type of image **:data**. If the **:data** is a list, it is assumed to be a **list** of **bitmaps** and an **image-xy** is created. If the **:data** is a **pixarray**, an **image-z** is created. Otherwise, the **:data** must be an array of bytes (**card8**), in which case a basic **image** object is created.

image-height *image* Function

Returns:
height — Type **card16**.

Returns the *height* of the *image* in pixels.

image — An **image** object.

image-name *image* Function

Returns:
name — Type **stringable** or **null**.

Returns and (with **setf**) changes the *name* string optionally associated with the *image*.

image — An **image** object.

image-plist *image* Function

Returns:
plist — Type **list**.

Returns and (with **setf**) changes the *image* property list. The property list is a hook for added application extensions.

image — An **image** object.

image-red-mask *image* Function

Returns:
mask — Type **pixel** or **null**.

Returns (and with **setf**) changes the *mask* which selects the pixel subfield for red intensity values. The *mask* is non-**nil** only for images for **:direct-color** or **:true-color** visual types.

image — An **image** object.

image-width *image* Function

Returns:
width — Type **card16**.

Returns the *width* of the *image* in pixels.

image — An **image** object.

image-x-hot *image* Function

Returns:
x-position — Type **card16** or **null**.

Returns and (with **setf**) changes the x position of the hot spot for an image used as a cursor glyph. The hot spot position is specified relative to the upper-left origin of the *image*.

image — An **image** object.

image-y-hot *image* Function

Returns:
y-position — Type **card16** or **null**.

Returns and (with **setf**) changes the y position of the hot spot for an image used as a cursor glyph. The hot spot position is specified relative to the upper-left origin of the *image*.

image — An **image** object.

7 IMAGES

Introduction

7.1 The X protocol provides for the transfer of images (two-dimensional arrays of pixel data) between a client program and a **drawable**. The format for image data can vary considerably. In order to present a uniform data representation for the manipulation of a variety of images, CLX defines a special **image** data type. Additional **image** subtypes — **image-xy** and **image-z** — allow for the representation of an image either as a sequence of bit planes or as an array of pixels. CLX includes functions for accessing **image** objects; for transferring image data between **image** objects, **drawables**, and files; and also for direct transfer of raw image data.

Image Types

7.2 The **image** data type is the base type for all **image** objects. **image-xy** and **image-z** are subtypes of the **image** type which furnish accessors specialized for different image representations.

Basic Images

7.2.1 The following paragraphs describe the CLX functions that can be used to access all types of **image** objects.

image-blue-mask *image* Function

Returns:
mask — Type **pixel** or **null**.

Returns (and with **setf**) changes the *mask* that selects the pixel subfield for blue intensity values. The *mask* is non-**nil** only for images for **:direct-color** or **:true-color** visual types.

image — An **image** object.

image-depth *image* Function

Returns:
depth — Type **card8**.

Returns the *depth* (that is, the number of bits per pixel) for the *image*.

image — An **image** object.

image-green-mask *image* Function

Returns:
mask — Type **pixel** or **null**.

Returns (and with **setf**) changes the mask that selects the pixel subfield for green intensity values. The mask is non-**nil** only for images for **:direct-color** or **:true-color** visual types.

image — An **image** object.