close-display *display* Function

Closes the connection to the X server for the specified *display*. It destroys all server resources (**window**, **font**, **pixmap**, **colormap**, **cursor**, and **gcontext**), that the client application has created on this display, unless the close down mode of the server resource has been changed (see **set-close-down-mode**). Therefore, these server resources should never be referenced again. In addition, this function discards any output requests that have been buffered but have not yet been sent.

display — A **display** object.

with-display *display* &body *body* Macro

This macro is for use in a multi-process environment. **with-display** provides exclusive access to the local **display** object for multiple request generation. It need not provide immediate exclusive access for replies. That is, if another process is waiting for a reply (while not in a **with-display**), then synchronization need not (but can) occur immediately. Except where noted, all routines effectively contain an implicit **with-display** where needed, so that correct synchronization is always provided at the interface level on a per-call basis. Nested uses of this macro work correctly. This macro does not prevent concurrent event processing (see **with-event-queue**).

display — A **display**.

Managing the Output Buffer

2.4 Most CLX functions cause output requests to be generated to an X server. Output requests are not transmitted immediately but instead are stored in an *output buffer* for the appropriate display. Requests in the output buffer are typically sent only when the buffer is filled. Alternatively, buffered requests can be sent prior to processing an event in the input event queue (see paragraph 12.3, Processing Events). In either case, CLX sends the output buffer automatically without explicit instructions from the client application.

However, in some cases, explicit control over the output buffer is needed, typically to ensure that the X server is in a consistent state before proceeding further. The **display-force-output** and **display-finish-output** functions allow a client program to synchronize with buffered output requests.

display-after-function *display* Function

Returns:
after-function — Type **function** or **null**.

Returns and (with **setf**) sets the *after-function* for the given *display*. If *after-function* is non-**nil**, it is a function that is called after every protocol request is generated, even those inside an explicit **with-display**, but never called from inside the *after-function*. The function is called inside the effective **with-display** for the associated request. The default value is **nil**. This can be set, for example, to #'**display-force-output** or #'**display-finish-output**.

display — A **display** object.

display-force-output *display* Function

Forces any buffered output to be sent to the X server.

display — A **display** object.

display-finish-output *display* Function

Forces any buffered output to be sent to the X server and then waits until all requests have been received and processed. Any errors generated are read and handled by the display error handler. Any events generated by output requests are read and stored in the event queue.

display — A **display** object.

Closing the Display

2.5 To close or disconnect a display from the X server, use **close-display**.

display-resource-id-mask *display* Function

Returns:
resource-id-mask — Type **resource-id**.

Returns the *resource-id-mask* that was returned from the server during connection setup for the specified *display*. The *resource-id-mask* contains a single contiguous set of bits (at least 18) which the client uses to allocate resource IDs for types **window**, **pixmap**, **cursor**, **font**, **gcontext**, and **colormap** by choosing a value with (only) some subset of these bits set, and **or**ing it with the *resource-id-base*. Only values constructed in this way can be used to name newly created server resources over this connection. Server resource IDs never have the top three bits set. The client is not restricted to linear or contiguous allocation of server resource IDs. Once an ID has been freed, it can be reused, but this should not be necessary.

An ID must be unique with respect to the IDs of all other server resources, not just other server resources of the same type. However, note that the value spaces of server resource identifiers, atoms, visualids, and keysyms are distinguished by context, and as such are not required to be disjoint (for example, a given numeric value might be both a valid window ID, a valid atom, and a valid keysym.)

display — A **display** object.

display-roots *display* Function

Returns:
roots — A list of screens.

Returns a list of all the **screen** structures available for the given *display*.

display — A **display** object.

display-vendor *display* Function

Returns:
vendor-name, *release-number* — Type **card32**.

Returns *vendor-name* and *release-number* as multiple values. See the **display-vendor-name** and **display-release-number** functions for additional information.

display — A **display** object.

display-vendor-name *display* Function

Returns:
vendor-name — Type **string**.

Returns a string that provides some vendor identification of the X server implementation associated with the specified *display*.

display — A **display** object.

display-version-number *display* Function

Returns:
version-number — Type **card16**.

Returns the X protocol version number for this implementation of CLX.

display — A **display** object.

display-xid *display* Function

Returns:
resource-allocator — Type **function**.

Returns the function that is used to allocate server resource IDs for this *display*.

display — A **display** object.

display-plist *display* Function

Returns:
plist — Type **list**.

Returns and (with **setf**) sets the property list for the specified *display*. This function provides a hook where extensions can add data.

display — A **display** object.

display-protocol-major-version *display* Function

Returns:
protocol-major-version — Type **card16**.

Returns the major version number of the X protocol associated with the specified *display*. In general, the major version would increment for incompatible changes. The returned protocol version number indicates the protocol the server actually supports. This might not equal the version supported by the client. The server can (but need not) refuse connections from clients that offer a different version than the server supports. A server can (but need not) support more than one version simultaneously.

display — A **display** object.

display-protocol-minor-version *display* Function

Returns:
protocol-minor-version — Type **card16**.

Returns the minor protocol revision number associated with the specified *display*. In general, the minor version would increment for small upward compatible changes in the X protocol.

display — A **display** object.

display-protocol-version *display* Function

Returns:
protocol-major-version, *protocol-minor-version* — Type **card16**.

Returns *protocol-major-version* and *protocol-minor-version* as multiple values. See the **display-protocol-major-version** and **display-protocol-minor-version** functions for additional information.

display — A **display** object.

display-resource-id-base *display* Function

Returns:
resource-id-base — Type **resource-id**.

Returns the *resource-id-base* value that was returned from the server during connection setup for the specified *display*. This is used in combination with the *resource-id-mask* to construct valid IDs for this connection.

display — A **display** object.

display — A **display** object.

display-max-keycode *display* Function

Returns:
max-keycode — Type **card8**.

Returns the maximum keycode value for the specified *display*. This value is never greater than 255. Not all keycodes in the allowed range are required to have corresponding keys.

display — A **display** object.

display-max-request-length *display* Function

Returns:
max-request-length — Type **card16**.

Returns the maximum length of a request, in four-byte units, that is accepted by the specified *display*. Requests larger than this generate a length error, and the server will read and simply discard the entire request. This length is always at least 4096 (that is, requests of length up to and including 16384 bytes are accepted by all servers).

display — A **display** object.

display-min-keycode *display* Function

Returns:
min-keycode — Type **card8**.

Returns the minimum keycode value for the specified *display*. This value is never less than eight. Not all keycodes in the allowed range are required to have corresponding keys.

display — A **display** object.

display-motion-buffer-size *display* Function

Returns:
motion-buffer-size — Type **card32**.

Returns the approximate size of the motion buffer for the specified *display*. The server can retain the recent history of pointer motion at a finer granularity than is reported by **:motion-notify** events. Such history is available through the **motion-events** function.

display — A **display** object.

display-p *display* Function

Returns:
display-p — Type **boolean**.

Returns non-**nil** if *display* is a **display** object; otherwise, returns **nil**.

display-pixmap-formats *display* Function

Returns:
pixmap-formats — Type **list**.

Returns the list of **pixmap-format** values for the given *display*. This list contains one entry for each depth value. The entry describes the format used to represent images of that depth. An entry for a depth is included if any screen supports that depth, and all screens supporting that depth must support (only) the format for that depth.

display — A **display** object.

display-error-handler *display* Function

Returns:
error-handler — Type **function** or **sequence**.

Returns and (with **setf**) sets the *error-handler* function for the given *display*. CLX calls (one of) the display error handler functions to handle server errors returned to the connection. The default error handler, **default-error-handler**, signals conditions as they occur. See Section 16, Errors, for a list of the conditions that CLX can signal. For more information about errors and error handling, refer to the section entitled Common Lisp Condition System in the *Lisp Reference* manual.

If the value of *error-handler* is a sequence, it is expected to contain a handler function for each specific error. The error code is used as an index into the sequence to fetch the appropriate handler function. If this element is a function, it is called for all errors. Any results returned by the handler are ignored since it is assumed the handler either takes care of the error completely or else signals. The arguments passed to the handler function are the **display** object, a symbol naming the type of error, and a set of keyword-value argument pairs that vary depending on the type of error. For all core errors, the keyword-value argument pairs are:

:current-sequence	**card16**
:major	**card8**
:minor	**card16**
:sequence	**card16**

For **colormap**, **cursor**, **drawable**, **font**, **gcontext**, **id-choice**, **pixmap**, and **window** errors, the keyword-value pairs are the core error pairs plus:

:resource-id	**card32**

For **:atom** errors, the keyword-value pairs are the core error pairs plus:

:atom-id	**card32**

For **:value** errors, the keyword-value pairs are the core error pairs plus:

:value	**card32**

display — A **display** object.

display-image-lsb-first-p *display* Function

Returns:
image-lsb-first-p — Type **boolean**.

Although the server is generally responsible for byte swapping communication data to match the client, images (pixmaps/bitmaps) are always transmitted and received in formats (including byte order) specified by the server. Within images for each scan-line unit in bitmaps or for each pixel value in pixmaps, the leftmost bit in the image as displayed on the screen is either the least or most significant bit in the unit. For the given *display*, **display-image-lsb-first-p** returns non-**nil** if the leftmost bit is the least significant bit; otherwise, it returns **nil**.

display — A **display** object.

display-keycode-range *display* Function

Returns:
min-keycode, *max-keycode* — Type **card8**.

Returns *min-keycode* and *max-keycode* as multiple values. See the **display-max-keycode** and **display-min-keycode** functions for additional information.

Display Attributes

2.3 The complete set of display attributes is discussed in the following paragraphs.

display-authorization-data *display* Function

Returns:
authorization-data — Type **string**.

Returns the authorization data string for *display* that was transmitted to the server by **open-display** during connection setup. The data is specific to the particular authorization protocol that was used. The **display-authorization-name** function returns the protocol used.

display — A **display** object.

display-authorization-name *display* Function

Returns:
authorization-name — Type **string**.

Returns the authorization protocol namestring for *display* that was transmitted by **open-display** to the server during connection setup. The *authorization-name* indicates what authorization protocol the client expects the server to use. Specification of valid authorization mechanisms is not part of the X protocol. A server that implements a different protocol than the client expects, or a server that only implements the host-based mechanism, can simply ignore this information. If both name and data strings are empty, this is to be interpreted as "no explicit authorization."

display — A **display** object.

display-bitmap-format *display* Function

Returns:
bitmap-format — Type **bitmap-format**.

Returns the *bitmap-format* information for the specified *display*.

display — A **display** object.

display-byte-order *display* Function

Returns:
byte-order — Either **:lsbfirst** or **:msbfirst**.

Returns the *byte-order* to be employed in communication with the server for the given *display*. The possible values are as follows:

- **:lsbfirst** — Values are transmitted least significant byte first.
- **:msbfirst** — Values are transmitted most significant byte first.

Except where explicitly noted in the protocol, all 16-bit and 32-bit quantities sent by the client must be transmitted with this *byte-order*, and all 16-bit and 32-bit quantities returned by the server are transmitted with this *byte-order*.

display — A **display** object.

display-display *display* Function

Returns:
display-number — Type **integer**.

Returns the *display-number* for the host associated with *display*.

display — A **display** object.

2 DISPLAYS

Introduction

2.1 A particular X server, together with its screens and input devices, is called a *display*. The CLX **display** object contains all the information about the particular display and its screens, as well as the state that is needed to communicate with the display over a particular connection.

Before your program can use a display, you must establish a connection to the X server driving your display. Once you have established a connection, you then can use the CLX macros and functions discussed in this section to return information about the display. This section discusses how to:

- Open (connect) a display
- Obtain information about a display
- Access and change display attributes
- Close (disconnect) a display

Opening the Display

2.2 The **open-display** function is used to open a connection to an X server.

open-display *host* &key **:display :protocol** Function

Returns:
display — Type **display**.

Returns a **display** that serves as the connection to the X server and contains all the information about that X server.

host — Specifies the name of the *host* machine on which the server executes. A string must be acceptable as a *host*, but otherwise the possible types are not constrained and will likely be very system dependent.

:display — An integer that specifies which display device on the *host* should be used for this connection. This is needed since multiple displays can be controlled by a single X server. The default is display 0 (zero).

:protocol — A keyword argument that specifies which network protocol should be used for connecting to the server (for example, **:tcp**, **:dna**, or **:chaos**). The set of possible values and the default value are implementation specific.

Authorization, if any, is assumed to come from the environment. After a successful call to **open-display**, all screens on the display can be used by the client application.